Red Hat JBoss Web Framework Kit 2.7
Start Developing

Tutorials for new users

Red Hat Customer Content Services

Red Hat JBoss Web Framework Kit 2.7 Start Developing

Tutorials for new users

Red Hat Customer Content Services

Legal Notice

Abstract

This document details how to get started with JBoss Web Framework Kit in Red Hat JBoss Developer Studio.

Table of Contents

Chapter 1. Introduction to Red Hat JBoss Web Framework Kit and Red Hat JBoss Developer Studio

1.1. About Red Hat JBoss Web Framework Kit

Red Hat JBoss Web Framework Kit is a set of enterprise-ready versions of popular open source frameworks. Together, these frameworks provide a solution for developing light and rich Java-based web applications.

JBoss Web Framework Kit comprises enterprise distributions of JBoss community web application frameworks and tested third-party frameworks. These leading frameworks support fast and easy client-side and server-side application development and testing, with frameworks including RichFaces, jQuery, jQuery Mobile, Hibernate Search, Spring, and Arquillian.

The breadth of JBoss Web Framework Kit frameworks provides choice for Java application development. Each framework has been tested and certified for use in applications deployed to Red Hat JBoss Enterprise Application Platform, Red Hat JBoss Web Server, or Red Hat OpenShift JBoss EAP cartridge. Inclusion in JBoss Web Framework Kit ensures stable versions of frameworks are available over long-term enterprise product life cycles, with regular releases for fixes and nonintrusive feature updates. Further, Red Hat JBoss Developer Studio (an Eclipse-based development environment) provides integrated project templates, quickstarts and tooling for many of the JBoss Web Framework Kit frameworks.

For the complete list of the frameworks composing JBoss Web Framework Kit and the certified platform and framework configurations, see https://access.redhat.com/site/articles/112543 on the Red Hat Customer Portal.

Report a bug

1.1.1. About the JBoss Web Framework Kit Tiers

The frameworks composing JBoss Web Framework Kit are categorized in four distinct tiers. A description of each tier and the associated Red Hat support is detailed here.

Tier 1 - Included Components

These are components that are based wholly or partly on open source technologies that support broad collaboration and where Red Hat maintains a leadership role; as such Red Hat is able to support these components and provide upgrades and fixes under our standard support terms and conditions.

Tier 2 - Tested Frameworks

These are third-party frameworks where Red Hat does not have sufficient influence and does not provide upgrades and fixes under our standard support terms and conditions. Commercially reasonable support is provided by Red Hat Global Support Services for these frameworks.

Tier 3 - Frameworks in Tested Examples

These are third-party frameworks where Red Hat does not have sufficient influence and does not provide upgrades and fixes under our standard support terms and conditions. Red Hat supports the examples these frameworks are used in and the generic use cases that these examples intend to demonstrate.

Tier 4 - Confirmed Frameworks

These are third-party frameworks that do not receive any support from Red Hat, but Red Hat verifies that the frameworks run successfully on Red Hat JBoss Enterprise Application Platform. Frameworks and versions not listed here have not been explicitly tested and certified, and thus may be subject to support limitations.

For a list of JBoss Web Framework Kit frameworks by tier, see https://access.redhat.com/site/articles/112543 on the Red Hat Customer Portal.

Report a bug

1.1.2. About the JBoss Web Framework Kit Distribution

The frameworks composing JBoss Web Framework Kit are distributed from a range of sources and in a variety of formats:

- The component frameworks are available from the Red Hat Customer Portal. They are distributed in two alternative formats: a binary for each framework or together as one Maven repository. In addition, the source code for each framework is provided for inspection.
- The third party frameworks are not distributed by Red Hat and each must be obtained from its own source.

A number of defined Maven JBoss stacks are provided as part of the JBoss Web Framework Kit distribution. All of the BOMs defining the JBoss stacks are available in the Maven repository **`.zip`** file available to download from the Red Hat Customer Portal or from http://www.jboss.org/developer-materials/ on the JBoss Developer Framework website.

An extensive set of examples are also provided as part of the JBoss Web Framework Kit distribution:

- TicketMonster is a moderately complex application demonstrating a number of the JBoss Web Framework Kit frameworks working together.
- Quickstarts and Maven archeypes illustrate subsets of the JBoss Web Framework Kit frameworks used to create simple applications.
- RichFaces, Snowdrop and Seam demonstrations showcase the power of each framework in web application development.

All of these examples are available from the Red Hat Customer Portal, with TicketMonster, the quickstarts, and the Maven archetypes also available from http://www.jboss.org/developer-materials/ on the JBoss Developer Framework website.

Report a bug

1.2. About Red Hat JBoss Developer Studio

Red Hat JBoss Developer Studio is a set of Eclipse-based development tools. It contains plug-ins that integrate with Eclipse to extend the existing functionality of the integrated development environment (IDE).

JBoss Developer Studio is designed to increase your productivity when developing applications. You can focus on building, testing and deploying your applications because JBoss application development tools are integrated in one IDE. Furthermore, JBoss Developer Studio can assist your application development with its unique features in the following ways:

- Develop new applications using the wizards and project examples of JBoss Central
- Add powerful functionality to applications with minimal effort using Forge Tools

- Build web interfaces with ease using the visual editing and drag-and-drop utilities of Visual Web Tools and Mobile Web Tools
- Experience browsers automatically refreshing in response to modified application resources with LiveReload Tools
- Incorporate JSF, Seam, JAX-RS, Hibernate, CDI and other popular APIs into applications with simplicity using the tool-driven interface
- Preview and test mobile web applications on a variety of simulation mobile devices using BrowserSim
- Create, build and test Cordova-based hybrid mobile applications for iOS and Android platforms using Hybrid Mobile Tools and CordovaSim
- Deploy applications to JBoss runtime servers and the cloud using JBoss Server Tools and OpenShift Tools

JBoss Developer Studio comes built around Eclipse and packaged with all of the necessary dependencies and third-party plug-ins for simplified installing. For developers already running Eclipse, JBoss Developer Studio can also be installed through Eclipse Marketplace. For distinction, this latter JBoss Developer Studio installation is called JBoss Developer Studio BYOE (Bring Your Own Eclipse).

Report a bug

1.2.1. Use Cases of JBoss Developer Studio

JBoss Developer Studio assists JEE developers by integrating JBoss technology and APIs in a single development environment. Here are a few ways that JBoss Developer Studio makes the work of a developer easier:

Web applications

JBoss Central provides wizards that generate skeleton and sample projects, enabling you to focus on developing the functionality of your applications. The wizards create web applications based on different APIs and technologies, showing the usage and advantages of each. JBoss Developer Studio also offers project file templates in a range of popular programming languages, including HTML, XHTML, and JSF.

Palettes in JBoss Developer Studio give access to the core elements of the JSF, Richfaces and Seam APIs, for use in developing the user interfaces of your applications. Elements of these APIs can be dragged and dropped directly into your project so that you can create richer user interfaces quickly. Visual Web Tools offers graphical and source viewing of files and defaults to dedicated editors for different file types. JBoss Developer Studio supports the Java EE specification and provides tools for JAX-RS, Hibernate, and CDI APIs so you can develop the server-side components of your applications effortlessly.

LiveReload Tools automatically refreshes browsers of local or deployed applications as you modify project resources to save you from needing to manually refresh. You can even experience automatic refreshing when viewing applications in browsers on external and mobile devices, with application web addresses easy to navigate to with QR codes.

Web applications optimized for mobile devices

Mobile Web Tools provides support for HTML5 and jQuery Mobile to enable you to create web applications optimized across desktop and mobile clients. The HTML5 Project wizard in JBoss Central generates a sample application using HTML5 and jQuery Mobile technologies and, together with HTML5 and jQuery Mobile project file templates, helps you

to get up and running with these APIs and technologies quickly. HTML5 and jQuery Mobile widgets can be dragged from the jQuery Mobile palette into your project files and, in conjunction with the widget wizards, enable you to effortlessly develop customized user interfaces for your mobile web applications.

BrowserSim allows you to view your web applications on a variety of simulated mobile devices so that you can ensure they will be correctly formatted. LiveReload also extends to BrowserSim allowing you to experience automatic browser refreshing as you develop your mobile web applications. The integration of Firebug Lite and Weinre capabilities with BrowserSim assists you to inspect the page source of web pages with familiar tools.

Hybrid mobile applications

Hybrid Mobile Tools provides support for developing and building Cordova-based hybrid mobile applications for iOS and Android platforms. The Hybrid Mobile application wizard assists you to quickly generate new projects, while the Cordova Configuration Editor and Cordova Plug-in Discovery wizard help you to efficiently modify the capabilities of your projects. Hybrid Mobile Tools provides actions that simplify your workflow, for example calling your system installed Android and iOS SDKs from within the IDE to emulate or run your hybrid mobile applications. With wizards to export workspace projects to Cordova-enabled native projects or ready-to-sign applications, you can quickly be ready to share your hybrid mobile projects and applications.

CordovaSim enables you to view and test your hybrid mobile applications on Android and iOS simulated mobile devices so that you can ensure they look and work as expected. You can interact with your mobile applications through BrowserSim and use the device input panel to provide sample data to your applications for device functions like cameras and accelerometers. An advantage of CordovaSim is that it does not require native SDKs to be installed on your system, unlike native SDK emulators. Additionally, by teaming the device control panel with BrowserSim, you get all of the great functionality of BrowserSim, such as skins and LiveReload, while simulating your hybrid mobile applications.

Applications for cloud deployment

OpenShift Tools deploys your applications directly to the cloud on the Red Hat OpenShift platform. You can create and manage your OpenShift account and manage the deployment of applications in your OpenShift domains from within the IDE. In addition to using the OpenShift Application wizard to create new applications for deployment to OpenShift, OpenShift Tools can import applications already deployed on OpenShift so that you can further develop them and manage their deployment from the comfort of the IDE.

Report a bug

1.3. Using JBoss Web Framework Kit in JBoss Developer Studio

JBoss Developer Studio offers an environment geared for developing, testing and deploying applications that use the JBoss Web Framework Kit frameworks, archetypes, quickstarts and examples.

Maven configuration for using the JBoss Web Framework Kit frameworks, JBoss Stacks and archetypes can be managed from within JBoss Developer Studio using JBoss Maven Integration and maven projects developed with the IDE Maven tools. Editors and palettes provide content assist and quick fixes for projects developed with the JBoss Web Framework Kit frameworks.

A number of the JBoss Web Framework Kit archetypes are available in JBoss Central under **Start from scratch** and from **File→New→Project**. The New Project wizards use the specified archetype to generate a new project, additionally enabling customization of the project name, package,

location and other project parameters. Note that the available archetype types and versions vary with releases of JBoss Developer Studio and JBoss Web Framework Kit; available archetypes have included jboss-html5-mobile-archetype (HTML5 Project), richfaces-archetype-kitchensink (RichFaces Project) and jboss-spring-mvc-archetype (Spring MVC Project).

A number of the JBoss Web Framework Kit quickstarts are available in JBoss Central under **`Start from a sample`**. Alternatively, you can import the JBoss Web Framework Kit quickstarts and examples into your IDE workspace using the IDE import wizards. The online TicketMonster tutorial can also be accessed from JBoss Central under **`Tutorial`**.

Furthermore, JBoss Developer Studio provides tooling for deploying applications to and managing JBoss Web Framework Kit certified versions of Red Hat JBoss Enterprise Application Platform and Red Hat OpenShift.

Report a bug

Chapter 2. Start Developing with JBoss Web Framework Kit

2.1. About Start Developing

Start Developing is a step-by-step guide that introduces JBoss Web Framework Kit and JBoss Developer Studio to new users. It aims to give you a taste of developing with JBoss Web Framework Kit in JBoss Developer Studio.

The guide includes a number of tutorials, taking JBoss Web Framework archetypes and quickstarts and completing the following tasks using JBoss Developer Studio:

- Create projects
- Deploy projects to servers
- View projects with browsers and mobile browser simulators
- Change project source code and republish applications

There are a number of general and tutorial-specific prerequisites that must be met in order to successfully complete the tutorials in this guide. These prerequisites are detailed in the appropriate guide sections.

Additionally, the guide describes one way to accomplish the tutorial tasks. However, there may be several alternative ways for each task and for achieving the overall result of each tutorial. For more information on using JBoss Web Framework Kit or JBoss Developer Studio, see the product documentation on the Red Hat Customer Portal.

Report a bug

2.2. Start Developing System Requirements

To ensure successful completion of the tutorials, your system must satisfy the following system requirements:

- JBoss Developer Studio 8.0 is installed and your system meets the associated system requirements
- Red Hat JBoss Enterprise Application Platform, version specific to this JBoss Web Framework Kit release, is installed and your system meets the associated system requirements
- JBoss EAP Maven repository, version matching installed JBoss EAP, is downloaded from the Red Hat Customer Portal and contents extracted
- JBoss Web Framework Kit Maven repository, version matching this release, is downloaded from the Red Hat Customer Portal and contents extracted

For information on the JBoss EAP version specific to this JBoss Web Framework Kit release, see https://access.redhat.com/site/articles/112543 on the Red Hat Customer Portal.

Report a bug

2.3. Start Developing Prerequisite Tasks

There are two prerequisite tasks that must be completed before commencing the tutorials. The JBoss Web Framework Kit projects in the Start Developing tutorials are Maven-based and require the JBoss

EAP and JBoss Web Framework Kit Maven repositories. Once downloaded and the contents extracted, you must configure Maven to use these repositories. Further, these tutorials deploy the project applications to JBoss EAP servers and so you must set up JBoss EAP servers for use within JBoss Developer Studio. These prerequisite tasks are detailed here.

Report a bug

2.3.1. Configure Maven

The JBoss Web Framework Kit projects in the Start Developing tutorials are Maven-based and require the JBoss EAP and JBoss Web Framework Kit Maven repositories. Maven is distributed with JBoss Developer Studio so it is not necessary to install Maven. But, once downloaded and the contents extracted, you must configure Maven to use these specific repositories by editing your Maven configuration file, settings.xml. You can edit settings.xml from within JBoss Developer Studio as detailed in the procedure below.

Important

The procedure details how to configure Maven for using the JBoss Web Framework Kit Maven repository. Once complete, you must repeat this procedure to configure Maven to also use the JBoss EAP Maven repository, replacing the file paths and names accordingly.

Procedure 2.1. Configure Maven

1. Click **Window**→**Preferences**, expand **JBoss Tools** and select **JBoss Maven Integration**.
2. Click **Configure Maven Repositories**.

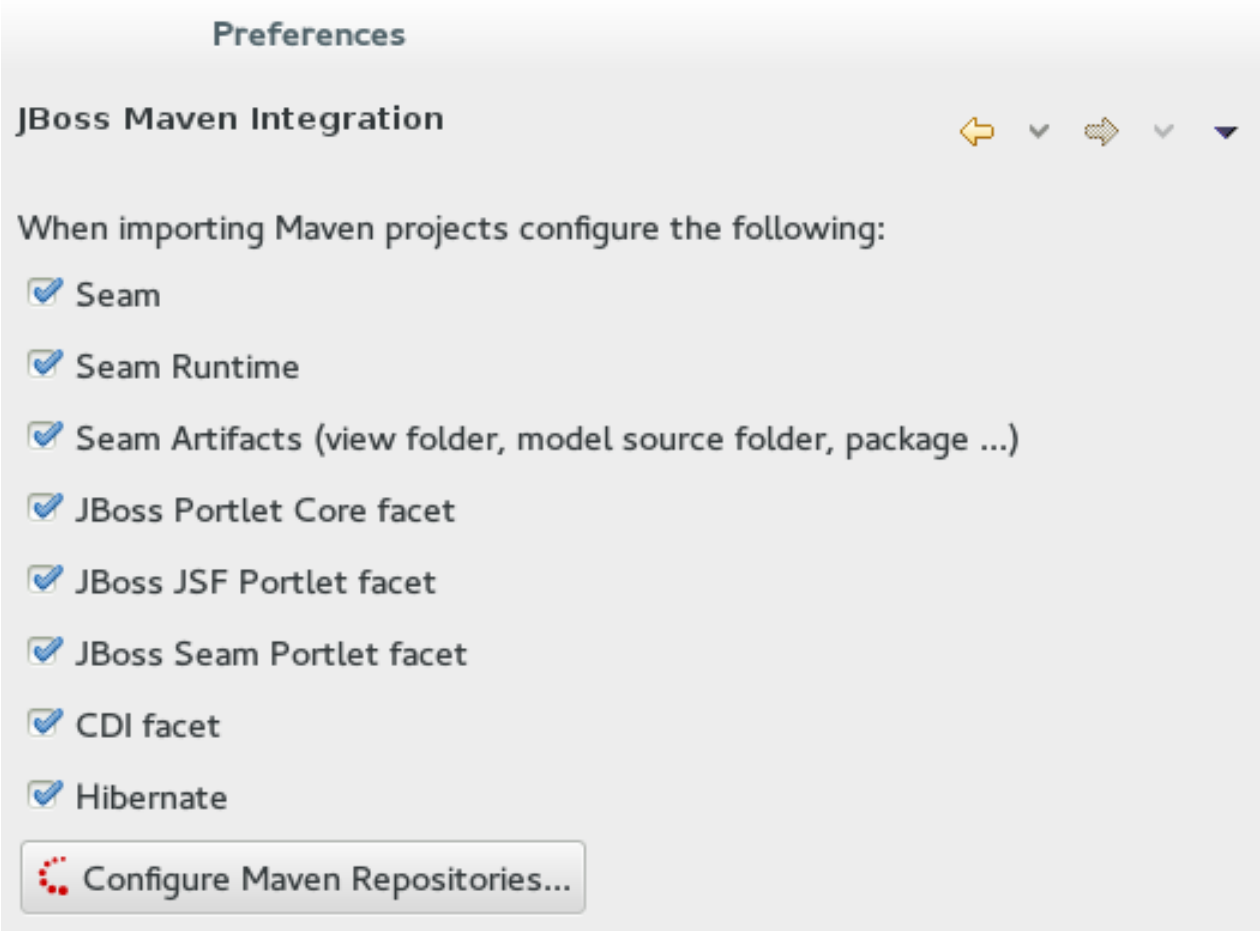

Figure 2.1. JBoss Maven Integration Pane in the Preferences Window

3. Click **`Add Repository`**.
4. Click **`Recognize JBoss Maven Enterprise Repositories`**.
5. Navigate to **`path/to/jboss-wfk-maven-repository`** and click **`OK`**. JBoss Maven Tools recursively scans the path searching for the Maven repository.
6. Modify the information in the **`ID`** and **`Name`** fields as desired, ensure the **`Active by default`** check box is selected and click **`OK`**.

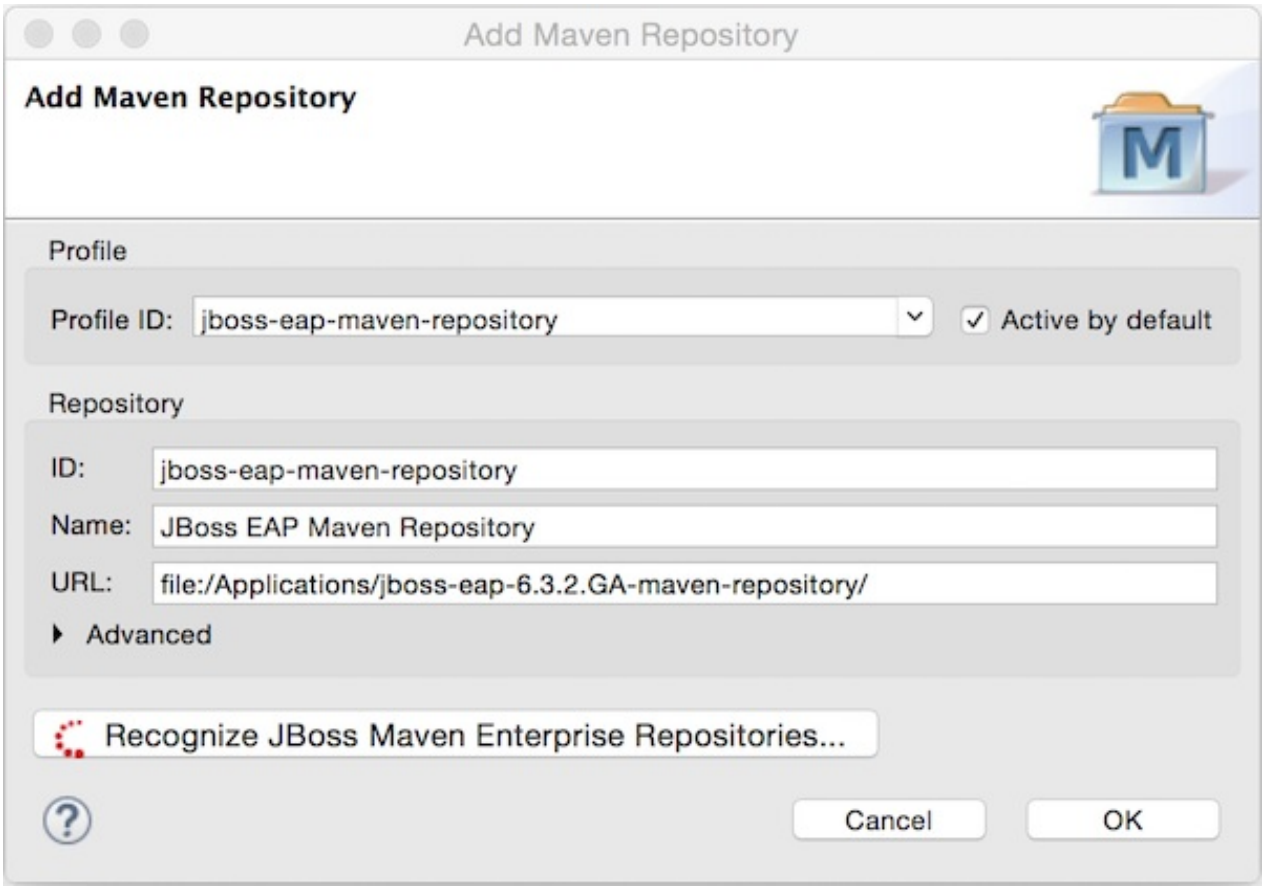

Figure 2.2. Add Maven Repository Wizard

7. Click **Finish** and at the prompt asking if you are sure you want to update the Maven configuration file click **Yes**. If the specified configuration file does not exist, JBoss Maven Tools creates it.
8. Click **Apply** and click **OK** to close the **Preferences** window.

Note

Maven settings, such as the configuration file, are specified in **Preferences** under **Maven→User Settings**. These settings can be customized.

Report a bug

2.3.2. Set Up a Server Using Runtime Detection

These tutorials deploy the project applications to JBoss EAP servers from within JBoss Developer Studio. To accomplish this, you must first inform the IDE about the servers you want to use. JBoss Server Tools provides a runtime detection feature and, as demonstrated in the procedure below, this locates and sets up servers so that they can be used from within the IDE.

Procedure 2.2. Set Up a Server Using Runtime Detection

1. Click **Window→Preferences**, expand **JBoss Tools** and select **JBoss Runtime Detection**.

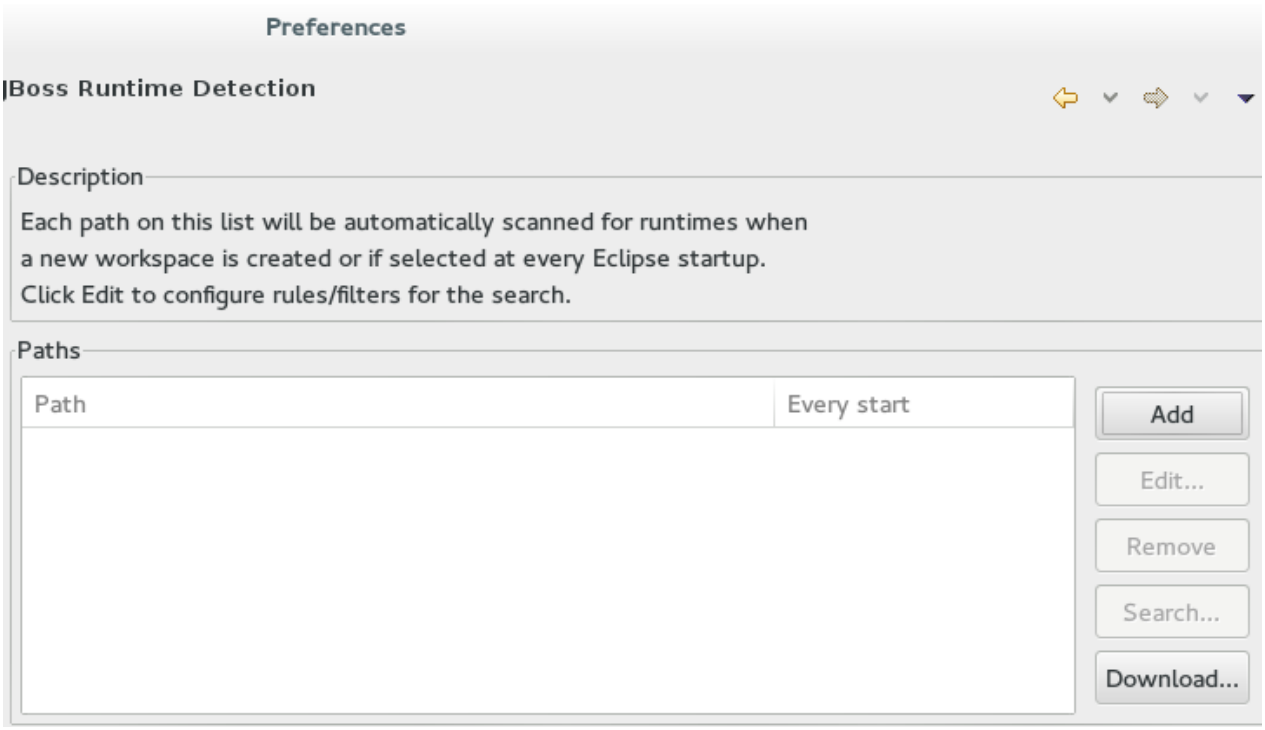

Figure 2.3. JBoss Runtime Detection Pane

2. Click **Add**.
3. Navigate to **path/to/jboss-eap** and click **OK**. JBoss Server Tools recursively scans the path searching for installed servers and displays a list of those it finds.
4. Ensure the **jboss-eap-*version*** check box is selected, where *version* denotes the JBoss EAP version, and click **OK**.

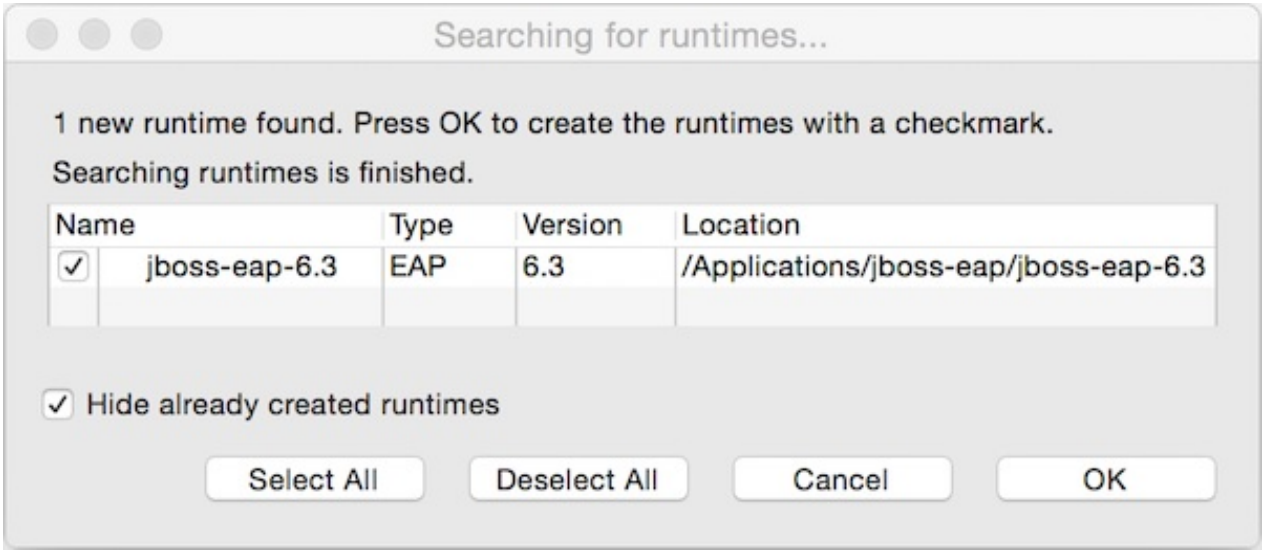

Figure 2.4. JBoss EAP 6.x Server Selected

5. Click **Apply** and click **OK** to close the **Preferences** window. The server is listed in the **Servers** view in stopped mode.

Report a bug

Chapter 3. HTML5 Archetype Tutorial

The HTML5 archetype is a Maven archetype that is released to Maven Central with this JBoss Web Framework Kit release. The archetype generates a simple HTML5 web application optimized for mobile devices, allowing you to register and list members. This tutorial provides an overview of the archetype and demonstrates how the archetype can be called from JBoss Developer Studio and subsequently changed, tested, and deployed using IDE tools.

Report a bug

3.1. About the HTML5 Archetype

The HTML5 archetype is a Maven archetype that is released to Maven Central with this JBoss Web Framework Kit release. The archetype generates a project for a simple HTML5 web application. This deployable web application is optimized for mobile devices, allowing you to register and list members.

The generated Maven 3 project is a basic JavaEE 6 application using HTML5, jQuery Mobile, JAX-RS, CDI 1.0, EJB 3.1, JPA 2.0, and Bean Validation 1.0. The project includes a persistence unit and some sample persistence and transaction code to help you get familiar with database access in enterprise Java EE.

The application uses a pure HTML client that interacts with the application server via restful endpoints (JAX-RS). This application also uses some of the latest HTML5 features and advanced JAX-RS. Testing is equally important for client-side as for server-side so this application uses QUnit to show you how to conduct unit tests for your JavaScript.

The application also integrates jQuery Mobile and basic client-side device detection to give you both a desktop and mobile version of the interface. Both versions support the same features, including form validation and member registration. However, the mobile version adds a mobile layout, touch, and performance improvements needed to get you started with mobile web development.

Report a bug

3.2. Tutorial-specific Prerequisites

There are no additional prerequisites for this tutorial.

Report a bug

3.3. Tutorial Overview

This tutorial demonstrates how to create, develop and deploy the HTML5 archetype using JBoss Developer Studio:

- Create the HTML5 Project from JBoss Central
- Deploy the application to JBoss EAP with JBoss Server Tools
- View the application with BrowserSim
- Enable LiveReload functionality in BrowserSim
- Modify the **`index.html`** file with JBoss Tools HTML Editor

- Modify the **Member.java** file
- Republish the application to JBoss EAP with JBoss Server Tools

You must work through these tasks in the order they are presented because the earlier tasks are prerequisites for the later tutorial tasks.

Report a bug

3.4. Create the HTML5 Project

The HTML5 archetype can be generated within JBoss Developer Studio using either the HTML5 Project wizard in JBoss Central or the IDE New Maven Project wizard. The procedures below outline both methods but you only need to use one. You should start by using the HTML5 Project wizard and check the archetype version it is using before continuing. The archetype version is predetermined in this project wizard and it may not match the version of this JBoss Web Framework Kit release. If it does not match, you must use the New Maven Project wizard instead, in which you can specify the archetype version explicitly.

Procedure 3.1. Create the HTML5 Project using the JBoss Central Wizard

1. In **JBoss Central**, under **Start from scratch**, click **HTML5 Project**. The **New Project Example** wizard opens.

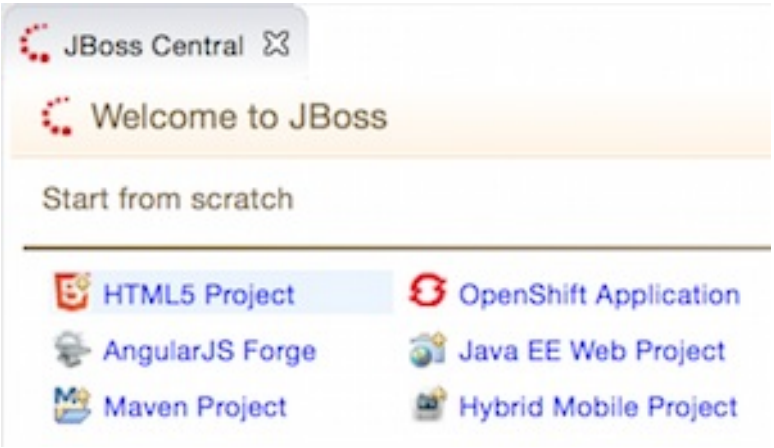

Figure 3.1. HTML5 Project in JBoss Central

2. From the **Target Runtime** list select **jboss-eap-*version* Runtime** where *version* denotes the JBoss EAP version.
3. Ensure that the **Description** field contains the archetype version matching the version of this JBoss Web Framework Kit release and click **Next**.
4. Complete the fields about the new project as follows:
   - In the **Project Name** field, type **my-mobile-app**.
   - In the **Package** field, type **com.company.example.mymobileapp**.

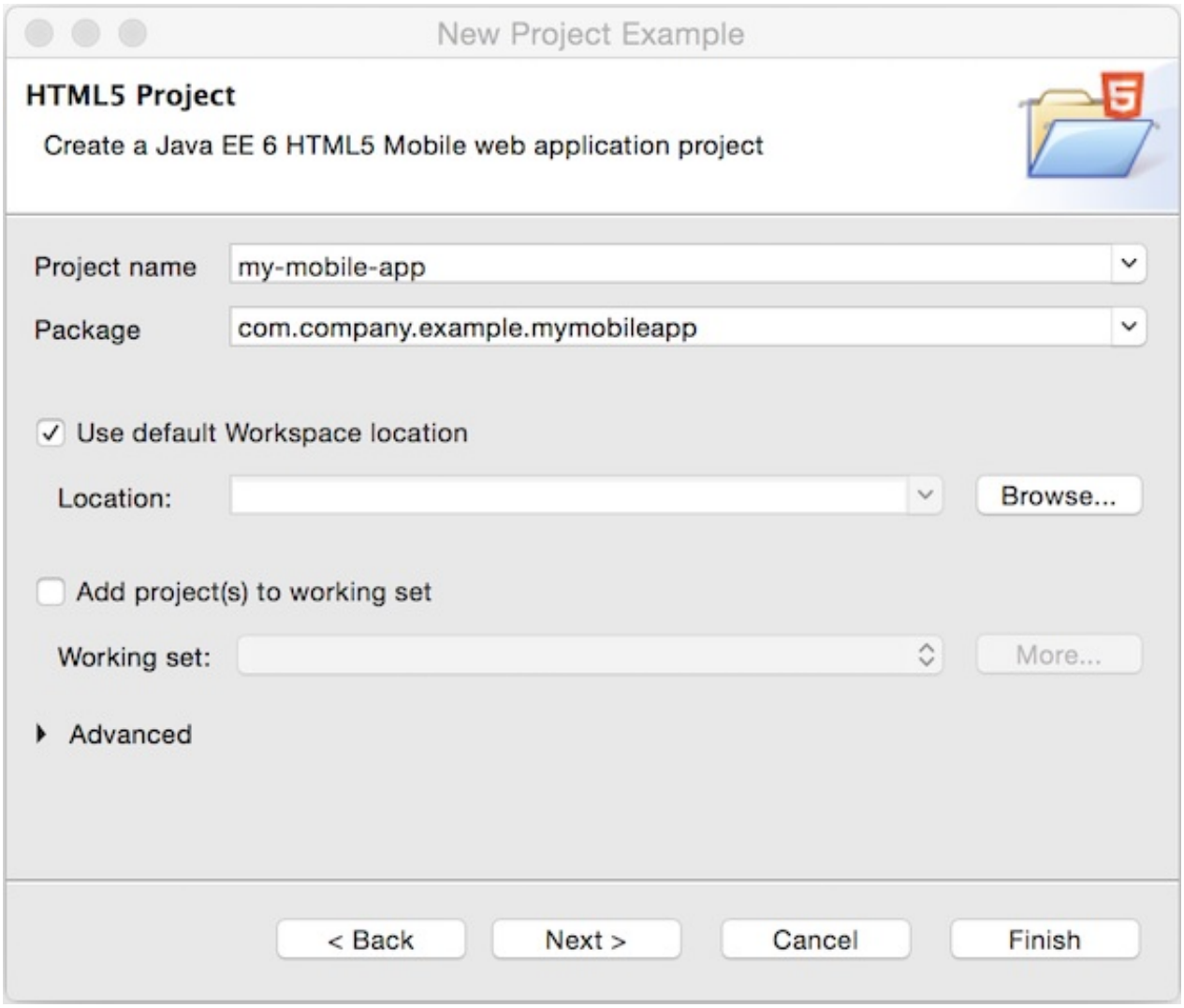

Figure 3.2. Completed Project Name and Package Fields in the New Project Example Wizard

5. Click **`Next`**.
6. Review the information you have entered and click **`Finish`**. Once the project is created, the **New Project Example** window opens and displays **`'HTML5 Project' Project is now ready`**.

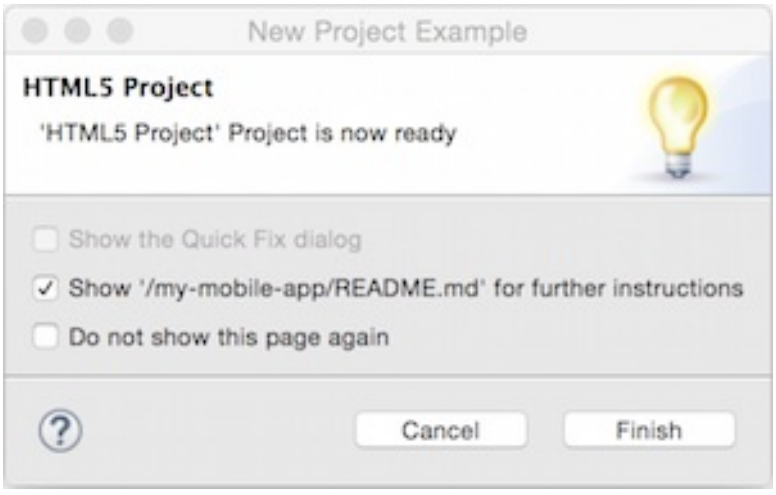

Figure 3.3. 'HTML5 Project' Project is now ready Message in the New Project

Example Wizard

7. Click **Finish** to close the **New Project Example** window. The project is listed in the **Project Explorer** view and a **README.md** file opens in a text editor displaying information about the project.

Procedure 3.2. Create the HTML5 Project using the New Maven Project Wizard

1. Click **File** → **New** → **Maven Project**.
2. Ensure the **Create a simple project** check box is not selected and click **Next**.

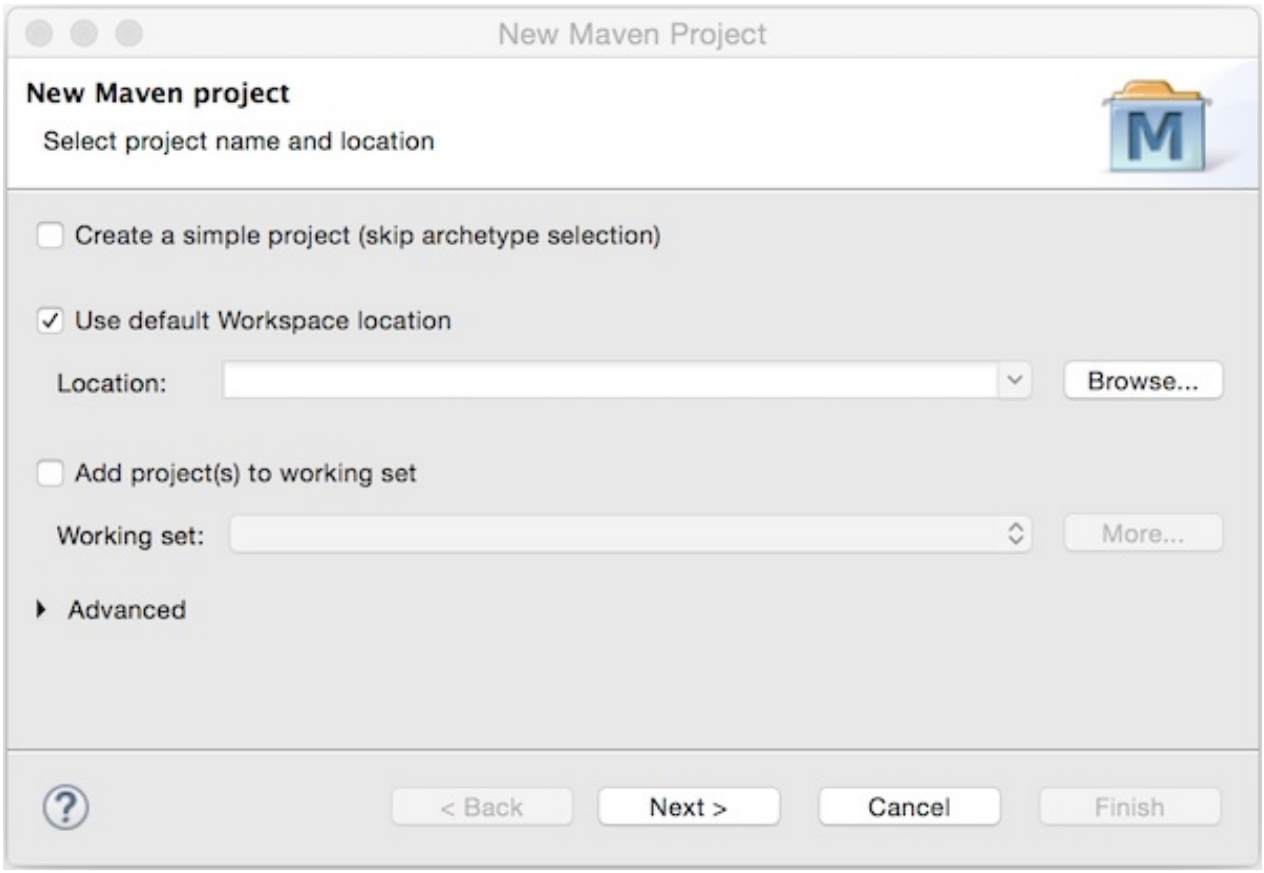

Figure 3.4. Create a simple project Check Box in the New Maven Project Wizard

3. Click **Add Archetype**.
4. Complete the fields about the JBoss Web Framework Kit HTML5 archetype as follows:
   - In the **Archetype Group Id** field, type **org.jboss.archetype.wfk**.
   - In the **Archetype Artifact Id** field, type **jboss-html5-mobile-archetype**.
   - In the **Archetype Version** field, type ***version*.Final** where *version* denotes the complete three-digit version number for this JBoss Web Framework Kit release. For example, **2.7.0.Final** for JBoss Web Framework Kit 2.7.0.
   - In the **Repository URL** field, type **http://central.sonatype.org/**.

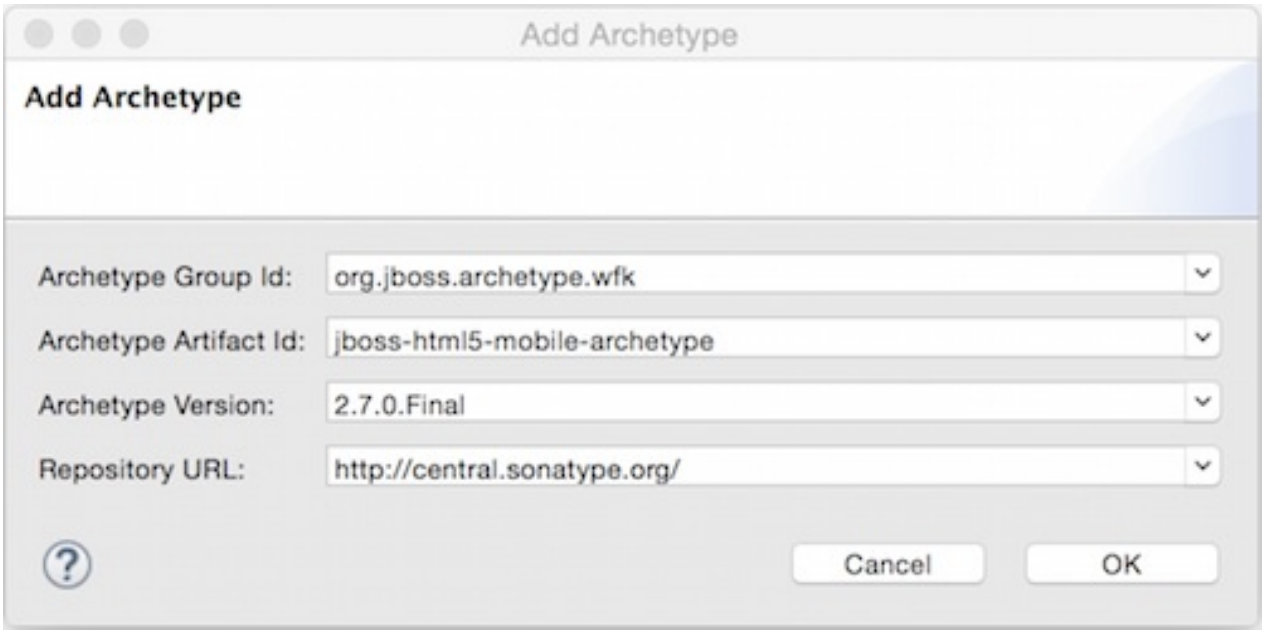

Figure 3.5. Completed Fields in the Add Archetype Window

5. Click **OK**.
6. From the archetype list, ensure the added archetype is selected and click **Next**.
7. Complete the fields about the project to be created as follows:
   - In the **Group Id** field, type **com.company.example.mymobileapp**.
   - In the **Artifact Id** field, type **my-mobile-app**.
8. Click **Finish**. Once the project is created, it is listed in the **Project Explorer** view.

Report a bug

3.5. Deploy and View the Application

3.5.1. Deploy the Application to the JBoss EAP Server

Once the my-mobile-app project is created, the project application can be deployed to the JBoss EAP server and the running application viewed in the default IDE web browser, as detailed in the procedure below.

Procedure 3.3. Deploy the Application to the JBoss EAP Server

1. In the **Project Explorer** view, right-click **my-mobile-app** and click **Run As** → **Run on Server**.
2. Under **How do you want to select the server?**, ensure **Choose an existing server** is selected.
3. In the **Server** table, expand **localhost**, select **jboss-eap-*version*** where *version* denotes the JBoss EAP version, and click **Next**.

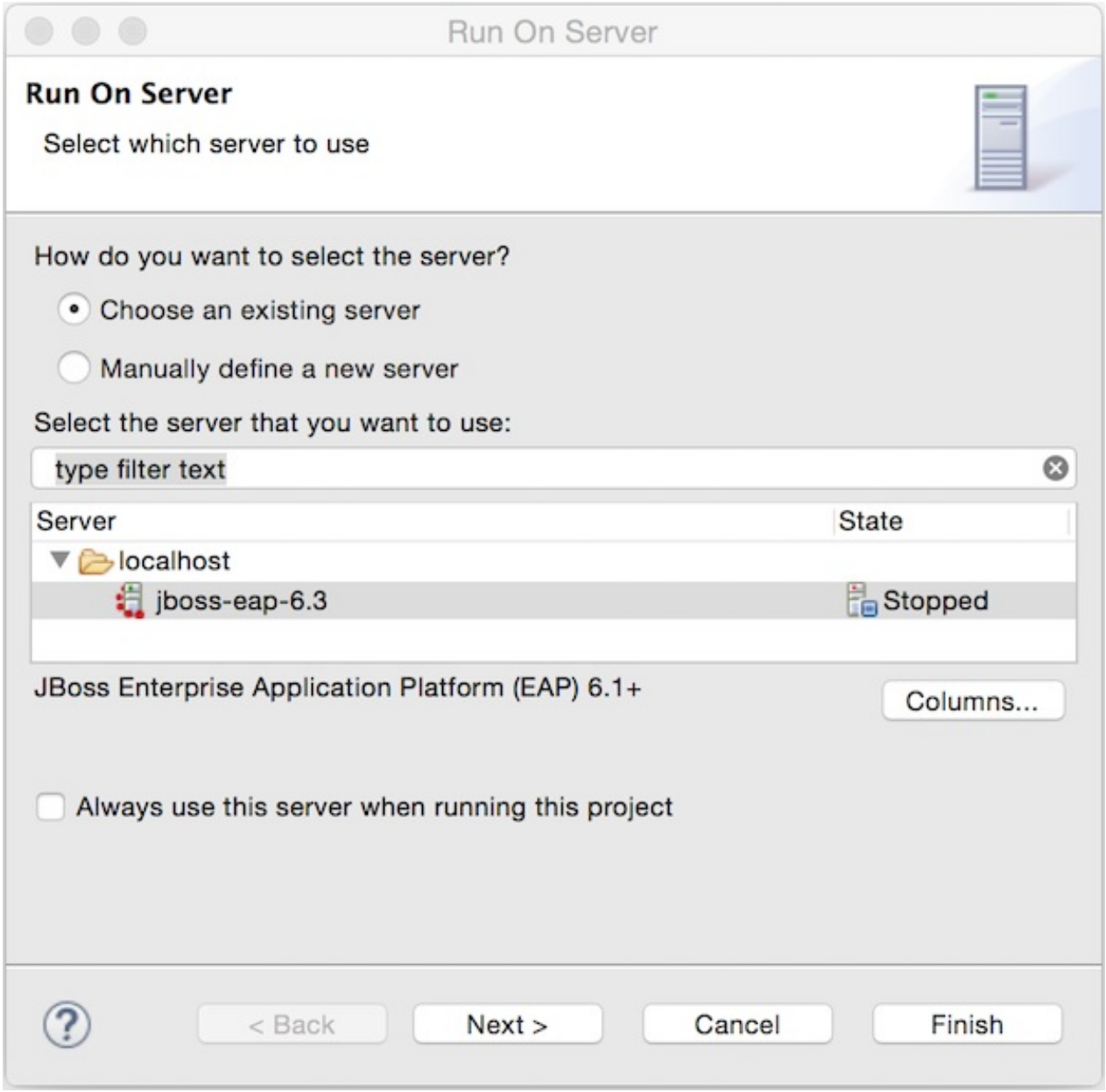

Figure 3.6. JBoss EAP 6.x Server Selected

4. Ensure that **my-mobile-app** is listed in the **Configured** column and click **Finish**. The **Console** view automatically becomes the view in focus and displays the output from the JBoss EAP server. Once deploying is complete, the web application opens in the default IDE web browser.

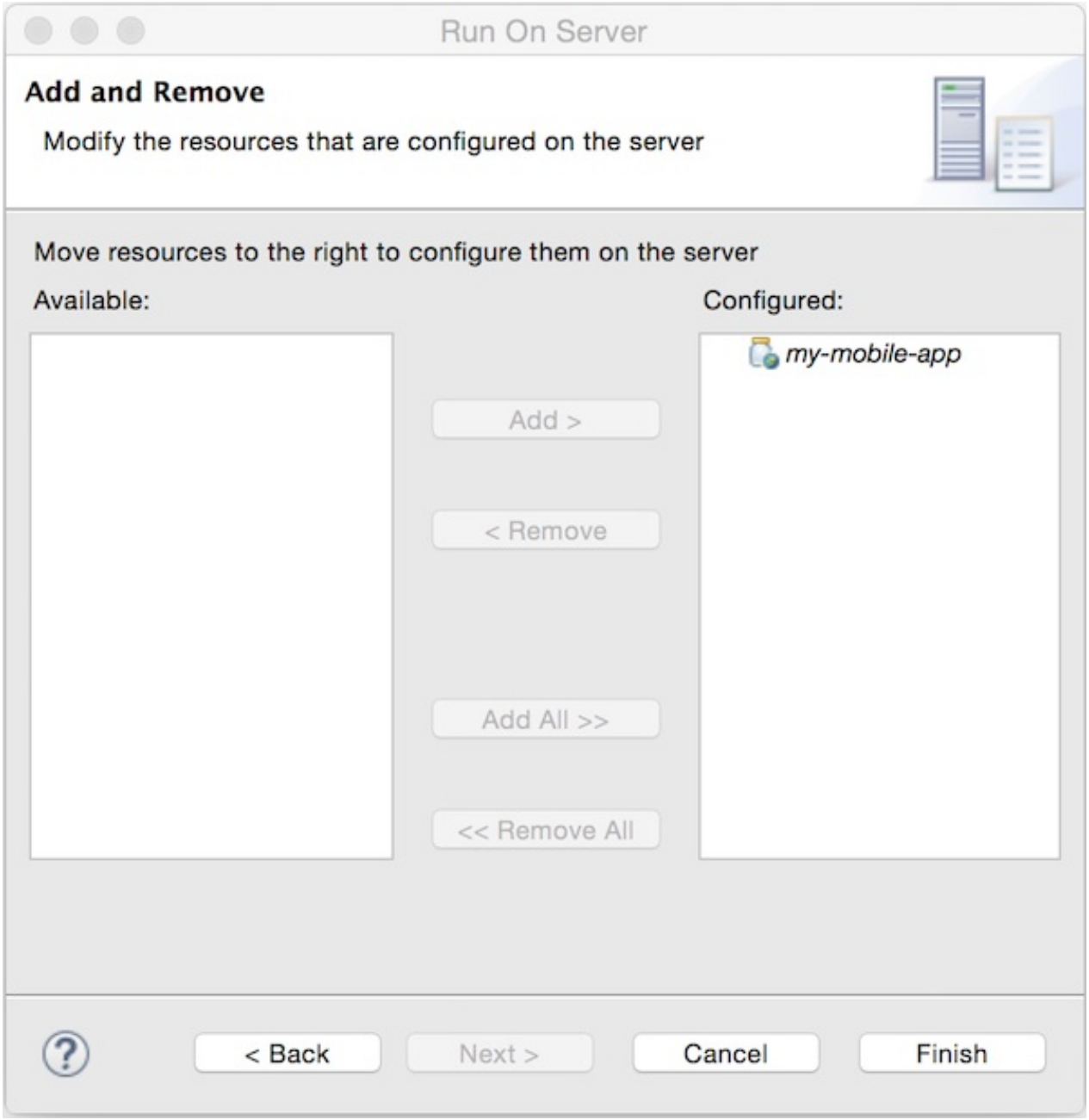

Figure 3.7. my-mobile-app Listed in the Configured Column

Report a bug

3.5.2. View the Application with BrowserSim

The my-mobile-app application is optimized for mobile devices and so it is best viewed on mobile devices. JBoss Developer Studio provides BrowserSim for this purpose; BrowserSim simulates web browsers of a variety of mobile devices. Details are provided here for viewing the my-mobile-app application with BrowserSim.

To view the deployed application with BrowserSim, in the **Servers** view, expand **jboss-eap-*version*** where *version* denotes the JBoss EAP version. Right-click **my-mobile-app**, and click **Show In** → **BrowserSim**. A BrowserSim simulated device opens displaying the application.

To rotate the simulated device, click any corner of the device. Alternatively, right-click the simulated device and click **Rotate Right** or **Rotate Left** as desired.

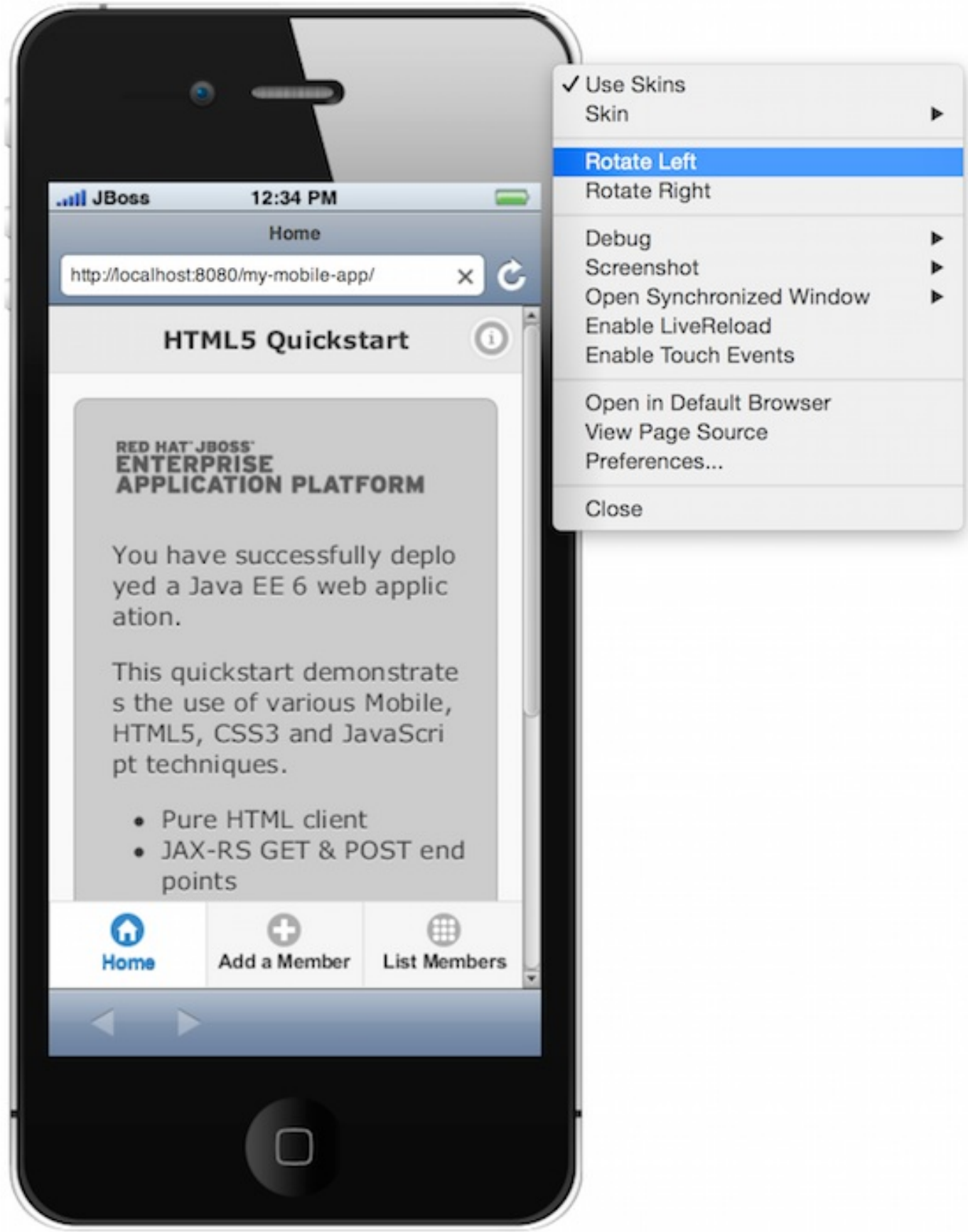

Figure 3.8. Rotating the Simulated Device

To view the application on a different BrowserSim simulated device, right-click the device and click **Skin**. From the list of skins, click the one with which you want to view the application.

Report a bug

3.6. Change the Application Client-side Functionality

3.6.1. Enable LiveReload in BrowserSim

Before making changes to the my-mobile-app project, it is useful to enable LiveReload in BrowserSim. LiveReload functionality automatically updates content displayed in web browsers as the corresponding source code is modified and saved in IDE editors, making the IDE workflow more efficient. This functionality is available for all web browsers and the procedure below details how to enable LiveReload in BrowserSim.

Procedure 3.4. Enable LiveReload in BrowserSim

1. In the **Servers** view, right-click **jboss-eap-*version*** where *version* denotes the JBoss EAP version, and click **New→Server**.
2. From the list of servers, expand **Basic**, select **LiveReload Server** and click **Finish**. The LiveReload server is listed in the **Servers** view.

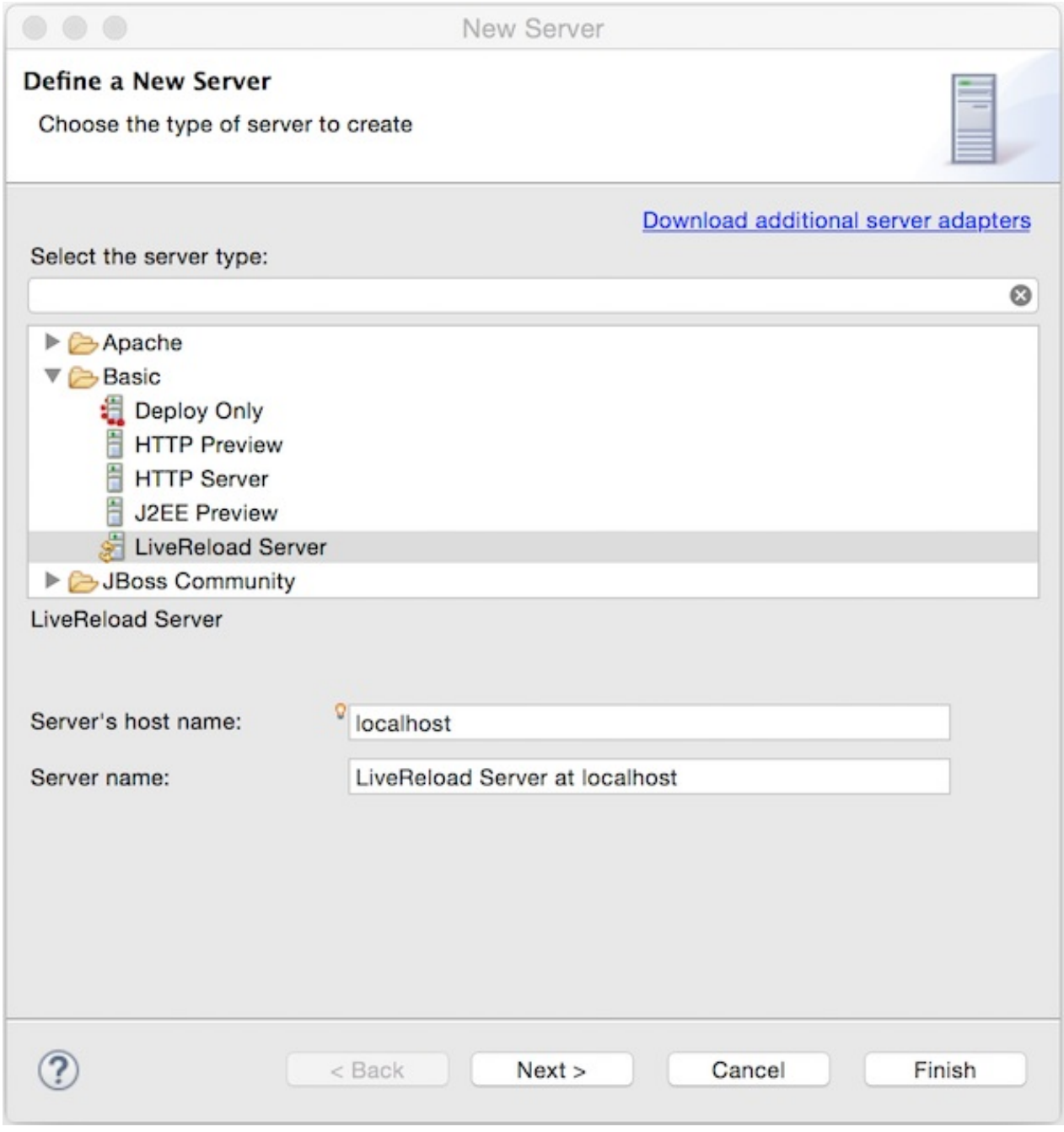

Figure 3.9. LiveReload Server Selected in the New Server Wizard

3. Right-click the **`LiveReload Server`** and click **Start**.
4. Right-click the BrowserSim simulated device and click **Enable LiveReload**. LiveReload functionality is now enabled on the simulated device and ready for use.

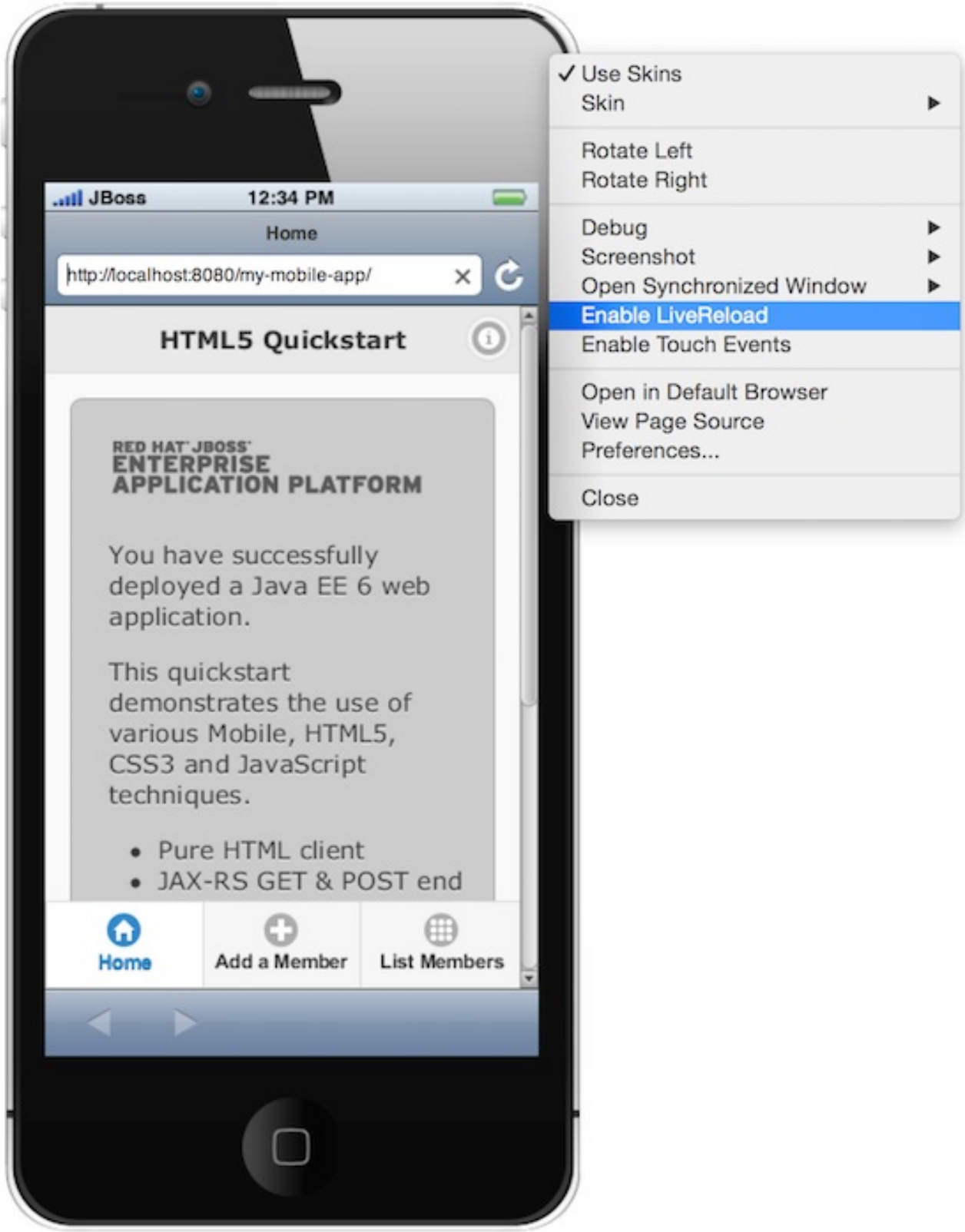

Figure 3.10. LiveReload Functionality Enabled on the Simulated Device

Report a bug

3.6.2. Modify `index.html`

The project HTML resources can be edited with the JBoss Tools HTML Editor, as demonstrated in the procedure below. With LiveReload enabled in BrowserSim, any changes to the HTML source are automatically reflected in the simulated device web browser on saving.

Procedure 3.5. Modify `index.html`

1. In the **`Project Explorer`** view, expand **`my-mobile-app`** → **`src`** → **`main`** → **`webapp`**.
2. Double-click **`index.html`** to open it in the JBoss Tools HTML Editor.
3. Ensure the JBoss Tools HTML Editor **`Source`** tab is the tab in focus.

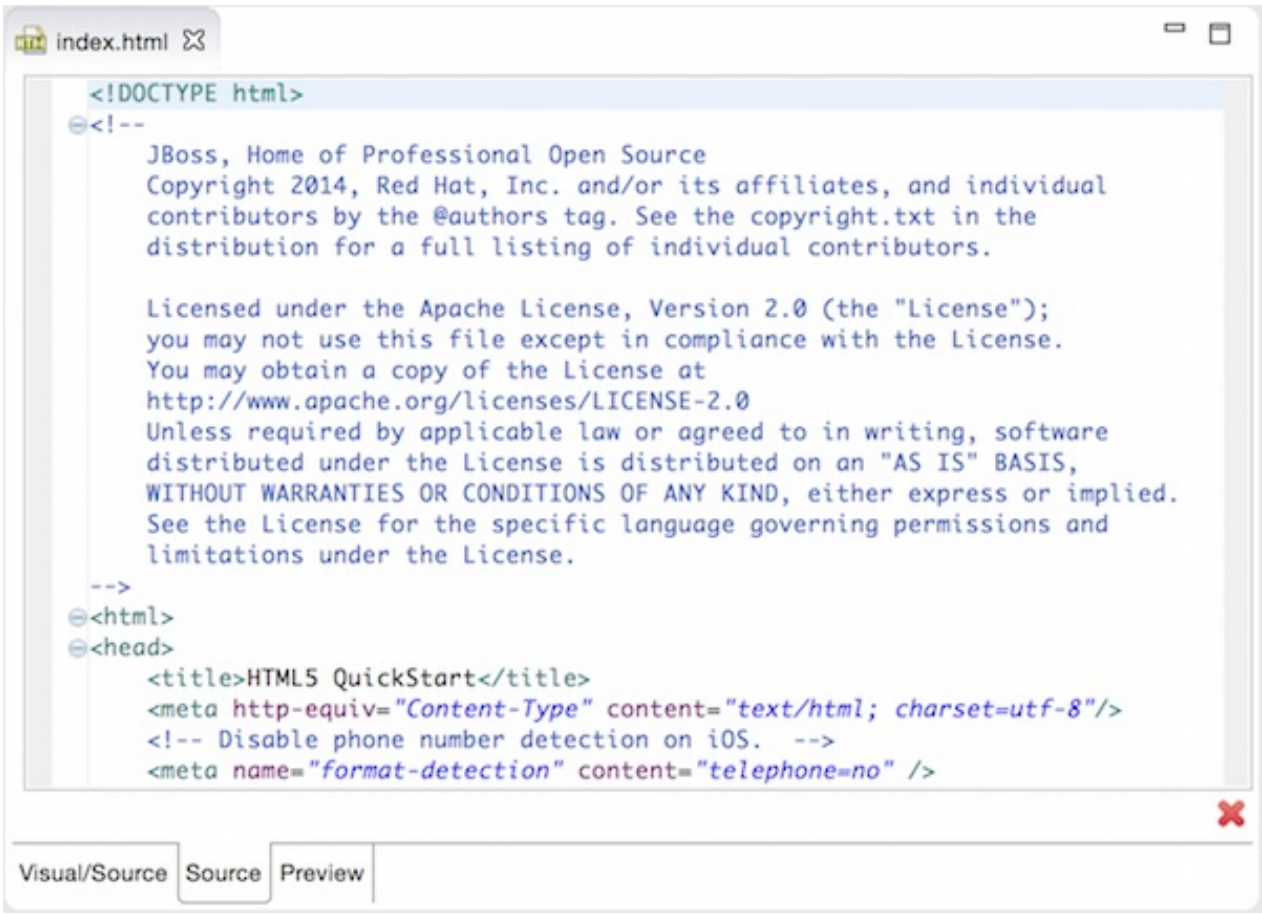

Figure 3.11. `index.html` Opened in the JBoss Tools HTML Editor

4. Scroll through the **`index.html`** file and locate the **`body`** tag.
5. Delete the following:

```
<p>You have successfully deployed a Java EE 6 web application.</p>
<p>This quickstart demonstrates the use of various Mobile, HTML5,
CSS3 and JavaScript techniques.</p>

<ul id="features">
<li class="feature">Pure HTML client</li>
```

```
<li class="feature">JAX-RS GET &amp; POST end points</li>
<li class="feature">HTML5 based page structure</li>
<li class="feature">HTML5 form element &amp; validation</li>
<li class="feature">CSS3 selectors used for styling</li>
<li class="feature">JAX-RS validation handling</li>
<li class="feature">jQuery Mobile integration</li>
<li class="feature">QUnit test suite to validate JavaScript</li>
</ul>
```

6. Replace the deleted code with the following:

```
<p>My Mobile Application!</p>
```

7. Save the **`index.html`** file by pressing **`Ctrl+S`**. Alternatively, to save click **File**→**Save** or click the **Save** icon. The page reloads in the BrowserSim simulated device, as a result of LiveReload being enabled, and shows the updated **`index.html`** file.

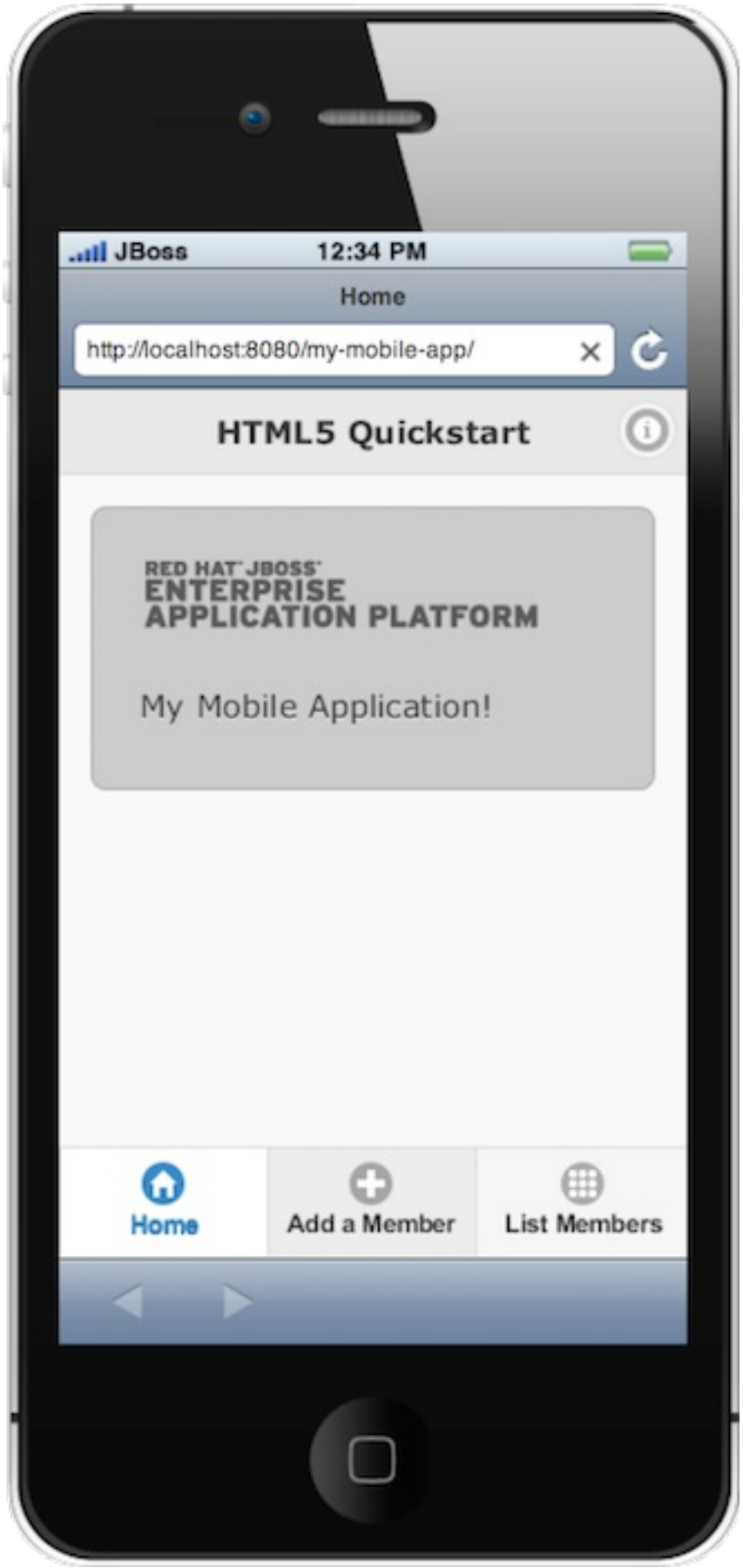

Figure 3.12. Changes to index.html File Shown on the Simulated Device

Report a bug

3.7. Change the Application Server-side Functionality

3.7.1. Modify Member.java

The Java resources can be edited with the IDE Java Editor, as demonstrated in the procedure below. Despite LiveReload being enabled in BrowserSim, these changes are not reflected until the application is republished to the server.

Procedure 3.6. Modify Member.java

1. In the **Project Explorer** view, expand **Java Resources**→**src/main/java**→**com.company.example.mymobileapp.model**.
2. Double-click **Member.java** to open it in the IDE Java Editor.

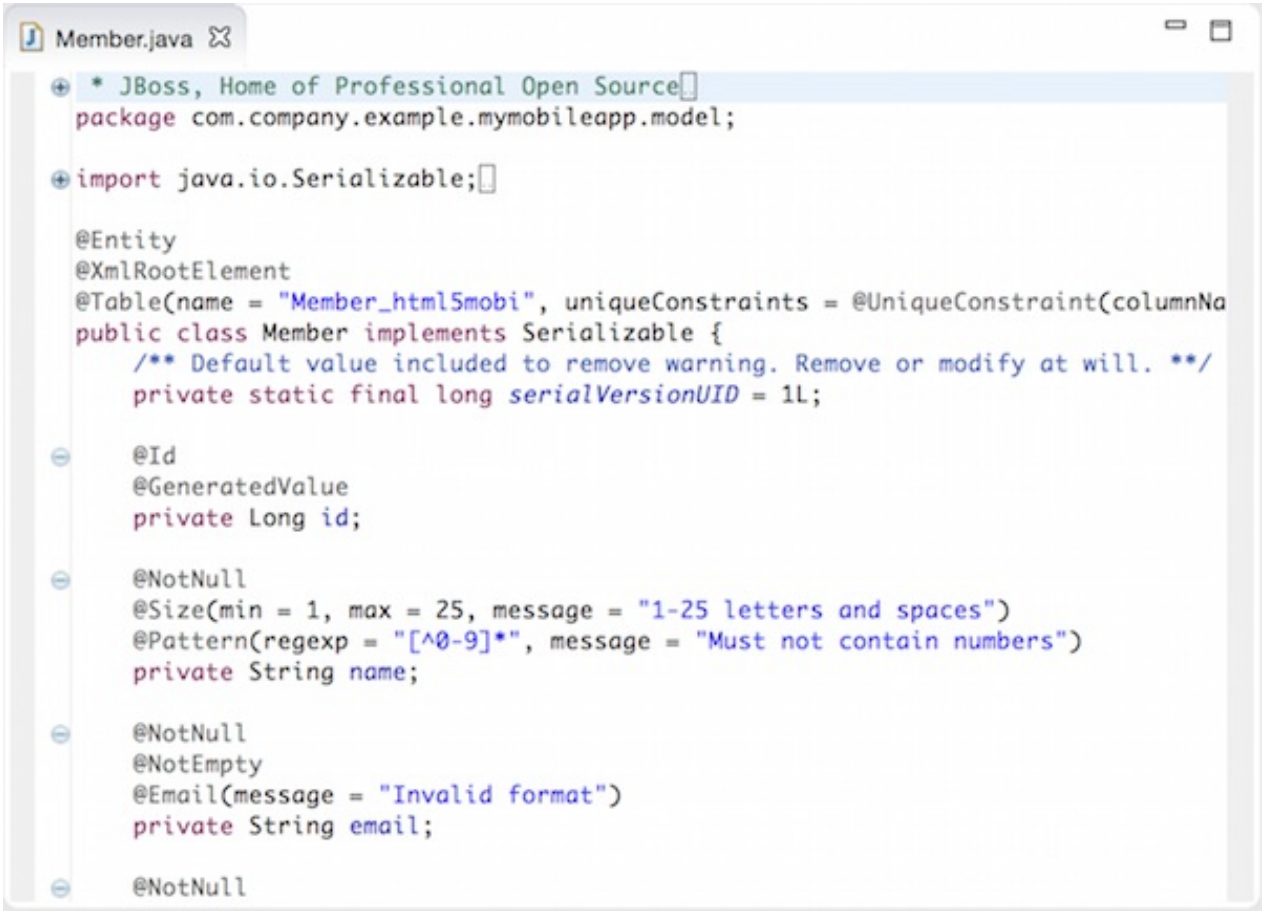

Figure 3.13. Member.java File Opened in the IDE Java Editor

3. Scroll through the **Member.java** file and locate the following code:

```
@NotNull
@Size(min = 10, max = 12, message = "10-12 Numbers")
@Digits(fraction = 0, integer = 12, message = "Not valid")
@Column(name = "phone_number")
private String phoneNumber;
```

4. Replace **min = 10** with **min = 6**.

5. Save the **Member.java** file.

Report a bug

3.7.2. Republish the Application

You must republish the application to the JBoss EAP server in order to see the **Member.java** changes reflected in the BrowserSim simulated device web browser, as detailed here.

To republish the application, in the **Servers** view, right-click **my-mobile-app** and click **Full Publish**. The JBoss EAP server republishes the application and, once complete, the **Console** view displays the following:

```
Replaced deployment "my-mobile-app.war" with deployment "my-mobile-
app.war"
```

To test the changes made to the **Member.java** file, in the simulated device web browser, in the **Phone #** field type a six digit number. You do not see the **phone # should be minimum 10 and maximum 12 digits** error message, indicating that you are viewing the updated running version of the application.

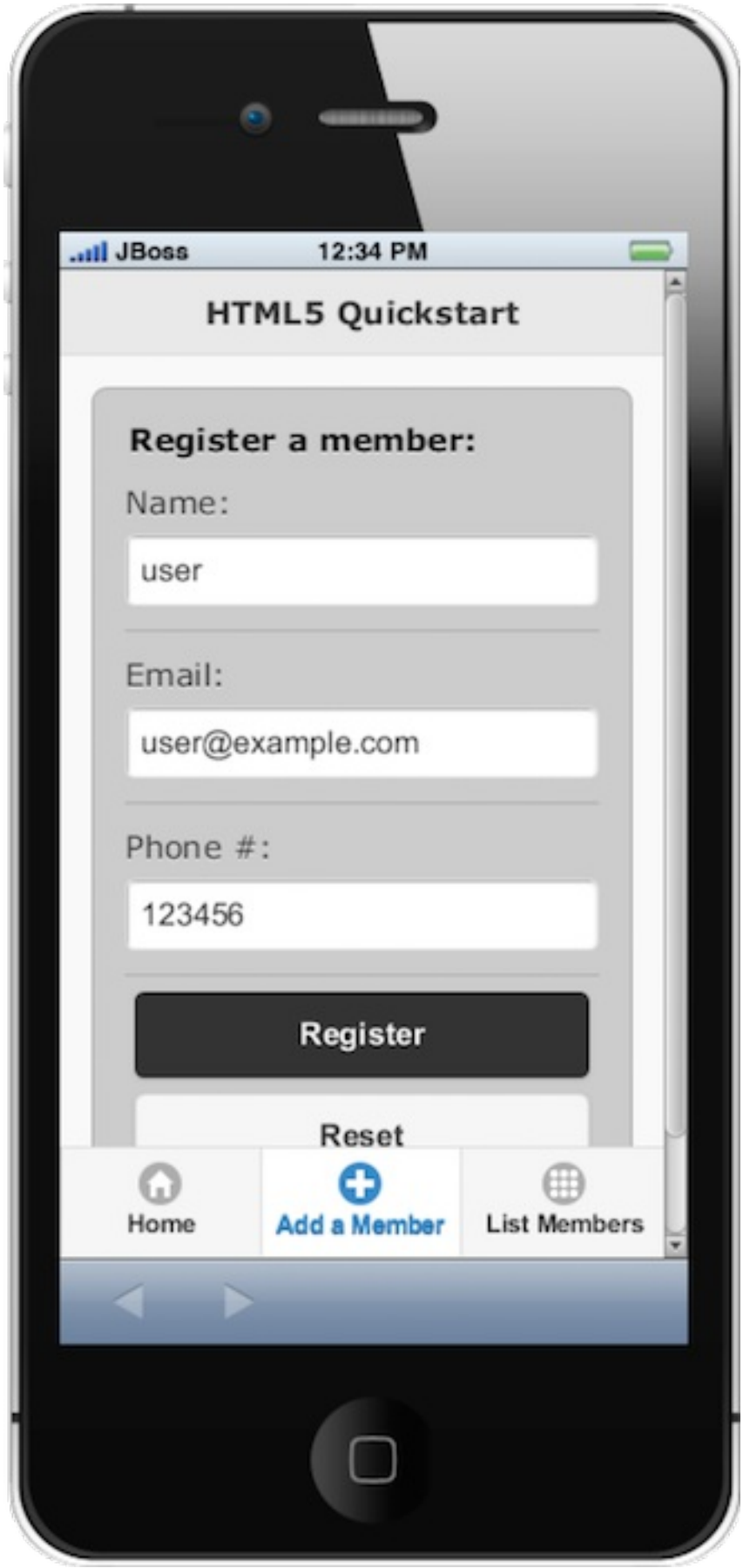

Figure 3.14. No Validation Message Shown for the Entered Six-digit Phone Number

Report a bug

Chapter 4. TicketMonster Example Tutorial

TicketMonster is an example web application that is distributed with each JBoss Web Framework Kit release. It is a simple application that shows many frameworks working in tandem to generate one application and to give the application a lot of functionality. This tutorial provides an overview of the application and demonstrates how the application can be imported into JBoss Developer Studio and subsequently changed, tested, and deployed using IDE tools.

Report a bug

4.1. About the TicketMonster Application

TicketMonster is an example web application demonstrating ticket purchasing. The application provides an online environment in which users can purchase tickets for events and administrators can manage the data relating to events and ticket sales. You can see a running example of the TicketMonster application at https://ticketmonster-jdf.rhcloud.com/.

The application consists of a user interface and web services. The user interface allows users to purchase tickets for listed events and it enables administrators to access and modify event and ticket information. The user interface is optimized for desktop and mobile clients. The web services enable access to information about members, events and venues stored in a database.

TicketMonster demonstrates the combining of Red Hat and JBoss technologies and frameworks to build, test and deploy applications. For example, the running instance of TicketMonster at https://ticketmonster-jdf.rhcloud.com/ is hosted in the cloud on the OpenShift platform.

Report a bug

4.2. Tutorial Overview

This tutorial demonstrates how to import, develop and deploy the TicketMonster example using JBoss Developer Studio:

- Obtain and import the TicketMonster example source code
- Deploy the application to JBoss EAP with JBoss Server Tools
- Deploy the application to OpenShift Online with OpenShift Tools
- View the application running on OpenShift in a web browser
- Modify the **`home.html`** file with JBoss Tools HTML Editor
- Redeploy the changed application to OpenShift Online with OpenShift Tools

You must work through these tasks in the order they are presented because the earlier tasks are prerequisites for the later tutorial tasks.

Report a bug

4.3. Tutorial-specific Prerequisites

This tutorial has additional JBoss Developer Studio prerequisites that must be completed before starting the tutorial as follows:

- Set the IDE default web browser to be your default system web browser. Click **Window→Web Browser→Default system web browser**.
- Set the Git connection timeout to 300 seconds. Click **Window→Preferences**, expand **Team** and select **Git**. In the **Remote connection timeout (seconds)** field, type **300** and click **Apply** and click **OK**.

Report a bug

4.4. Create the TicketMonster Project

4.4.1. Obtain the Project Source Code

The TicketMonster example source code is provided as a **.zip** file on the Red Hat Customer Portal. This procedure guides you through downloading this **.zip** file and extracting the contents ready for use with JBoss Developer Studio. The TicketMonster source code can also be downloaded from the JBoss Developer website.

Procedure 4.1. Obtain the Project Source Code

1. Log into the Red Hat Customer Portal at http://access.redhat.com.
2. Locate the downloads for this release version of JBoss Web Framework Kit.
3. Download the **.zip** file for Red Hat JBoss Web Framework Kit TicketMonster Demo.
4. Extract the contents of the downloaded **.zip** file to the location where you want to keep the TicketMonster source code.

Report a bug

4.4.2. Import the Project Source Code

Once the TicketMonster source code is obtained and unpackaged, you must import it into JBoss Developer Studio, as detailed in the procedure below. TicketMonster is a Maven-based project so a specific Import Maven Project wizard is used for the import.

Procedure 4.2. Import the Project Source Code

1. Click **File→Import** to open the **Import** wizard.
2. Expand **Maven**, select **Existing Maven Projects** and click **Next**.
3. In the **Root Directory** field, enter the path to the TicketMonster source code. Alternatively, click **Browse** to navigate to the source code location. The **Import Maven Project** wizard recursively searches the path for a **pom.xml** file. The **pom.xml** file identifies the project as a Maven project. The file is listed under **Projects** once it is found.

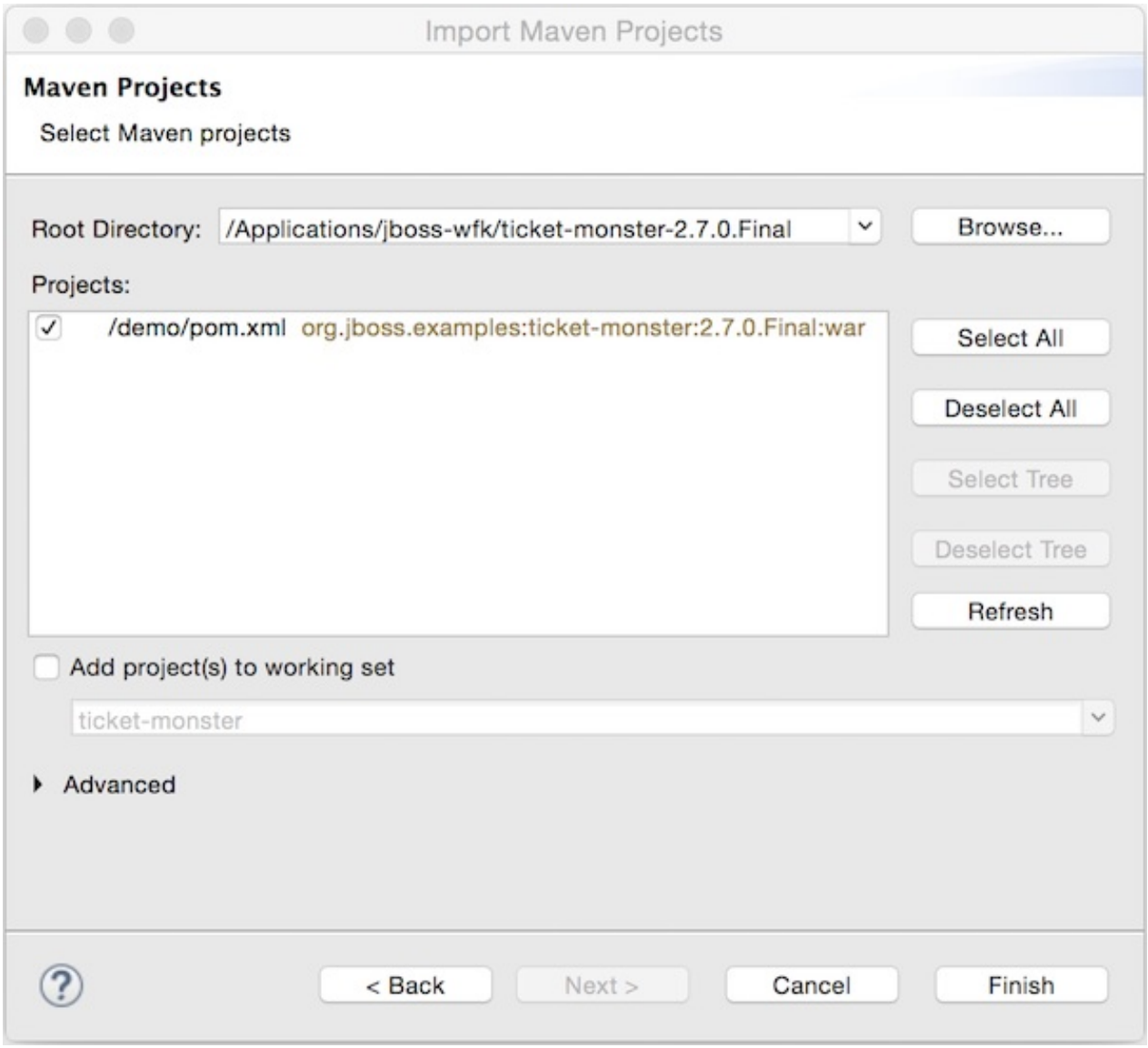

Figure 4.1. pom.xml File Listed in the Projects Pane

4. Click **Finish**. When the import process is complete, the project is listed in the **Project Explorer** view.

Report a bug

4.5. Deploy and View Application

4.5.1. Deploy to the JBoss EAP Server

Once you have imported the TicketMonster source code into JBoss Developer Studio, the project application can be deployed to the JBoss EAP server and the running application viewed in the default system web browser, as detailed in the procedure below.

Procedure 4.3. Deploy to the JBoss EAP Server

1. In the **Project Explorer** view, right-click **ticket-monster** and click **Run As** → **Run on Server**.

2. Under **How do you want to select the server?**, ensure **Choose an existing server** is selected.

3. In the **Server** table, expand **localhost**, select **jboss-eap-*version*** where *version* denotes the JBoss EAP version, and click **Next**.

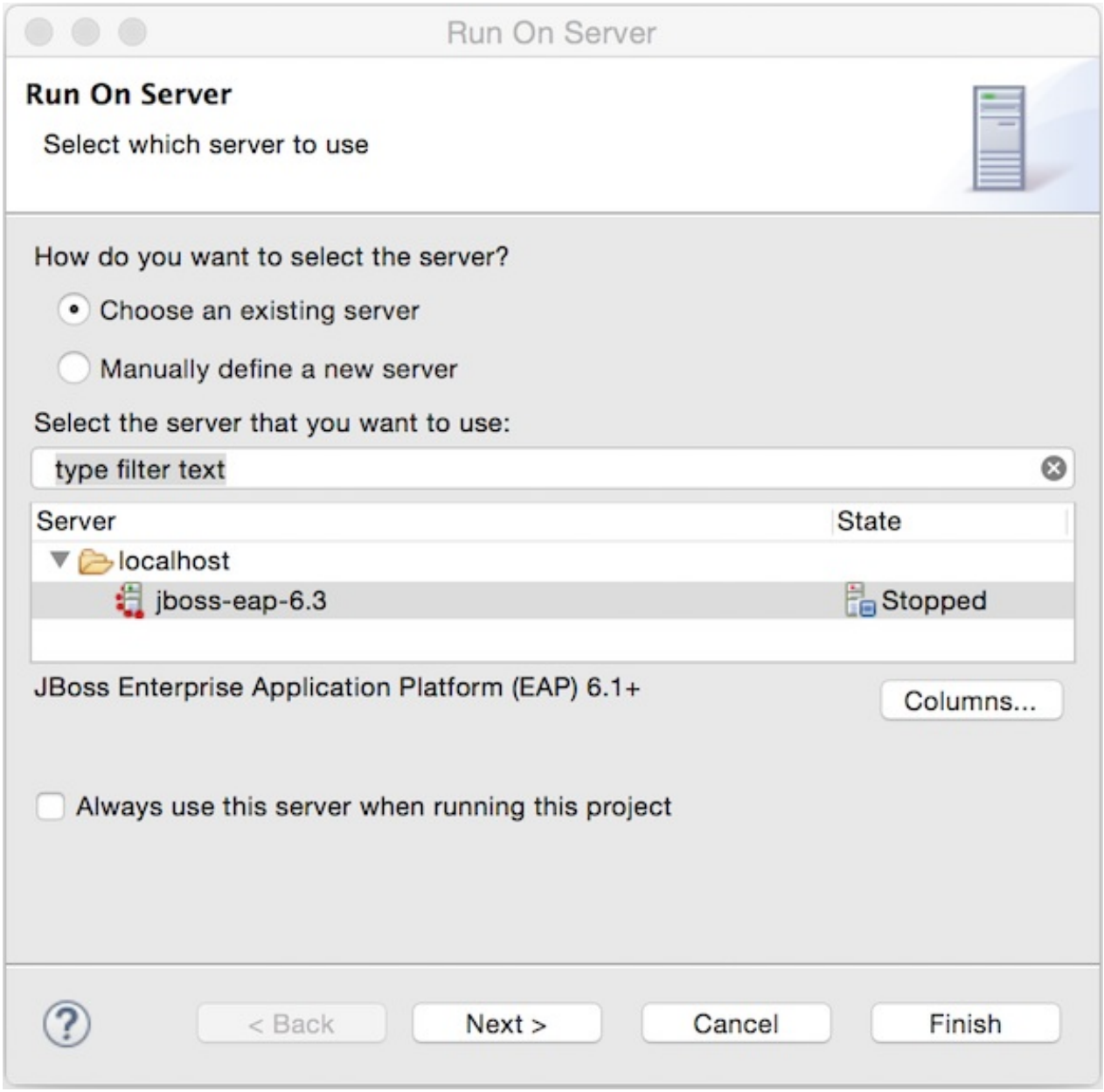

Figure 4.2. JBoss EAP 6.x Server Selected

4. Ensure **ticket-monster** is listed in the **Configured** column and click **Finish**. The **Console** view automatically becomes the view in focus and displays the output from the JBoss EAP server. Once deploying is complete, the web application opens in the default system web browser.

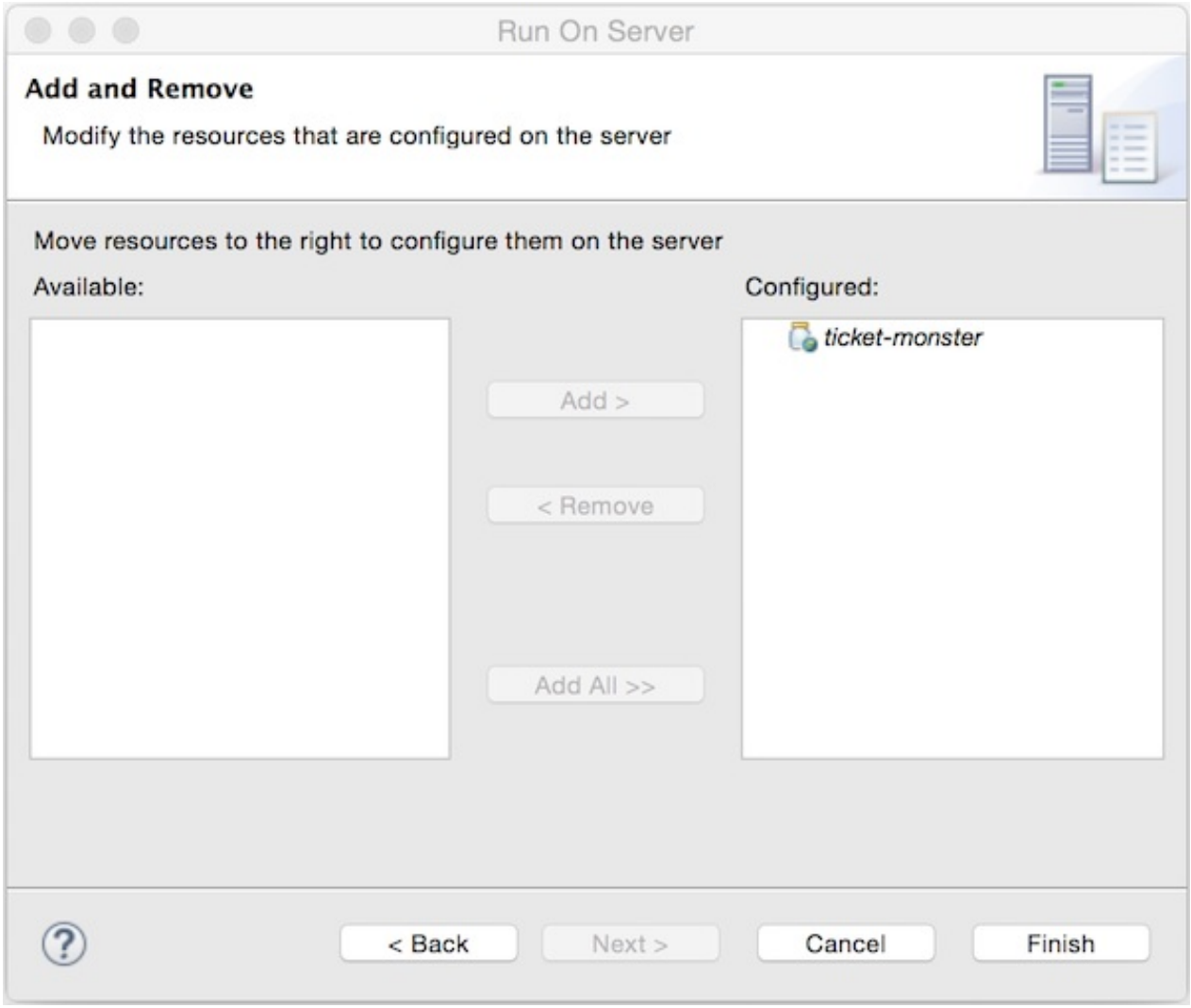

Figure 4.3. ticket-monster Listed in the Configured Column

Report a bug

4.5.2. Deploy to OpenShift Online

Once verified that the workspace ticket-monster application runs successfully on JBoss EAP, you can deploy it with confidence to OpenShift Online, Red Hat's cloud service. To deploy ticket-monster to OpenShift Online, you must create a new OpenShift Online application based on the existing workspace project using OpenShift Tools, as detailed in the procedure below.

Note

This procedure documents the deploying process for first-time OpenShift Online users. This includes one-time steps, such as signing up for an OpenShift Online account, creating an OpenShift Online domain and uploading SSH keys. If you have previously used OpenShift Online and OpenShift Tools, omit the one-time steps as appropriate.

Procedure 4.4. Deploy to OpenShift Online

1. In **`JBoss Central`**, under **`Start from scratch`**, click **`OpenShift Application`**.

2. If you do not have an OpenShift Online account, click **Please sign up here** to create an OpenShift Online account and follow the instructions on the OpenShift web page displayed in your default system web browser. Once you have completed the sign-up process, restart the **New OpenShift Application** wizard.
3. Complete the fields about your OpenShift Online account as follows:
   - From the **Connection** list, select **New Connection**.
   - Ensure the **Use default server** check box is selected.
   - In the **Username** and **Password** fields, type your account credentials.

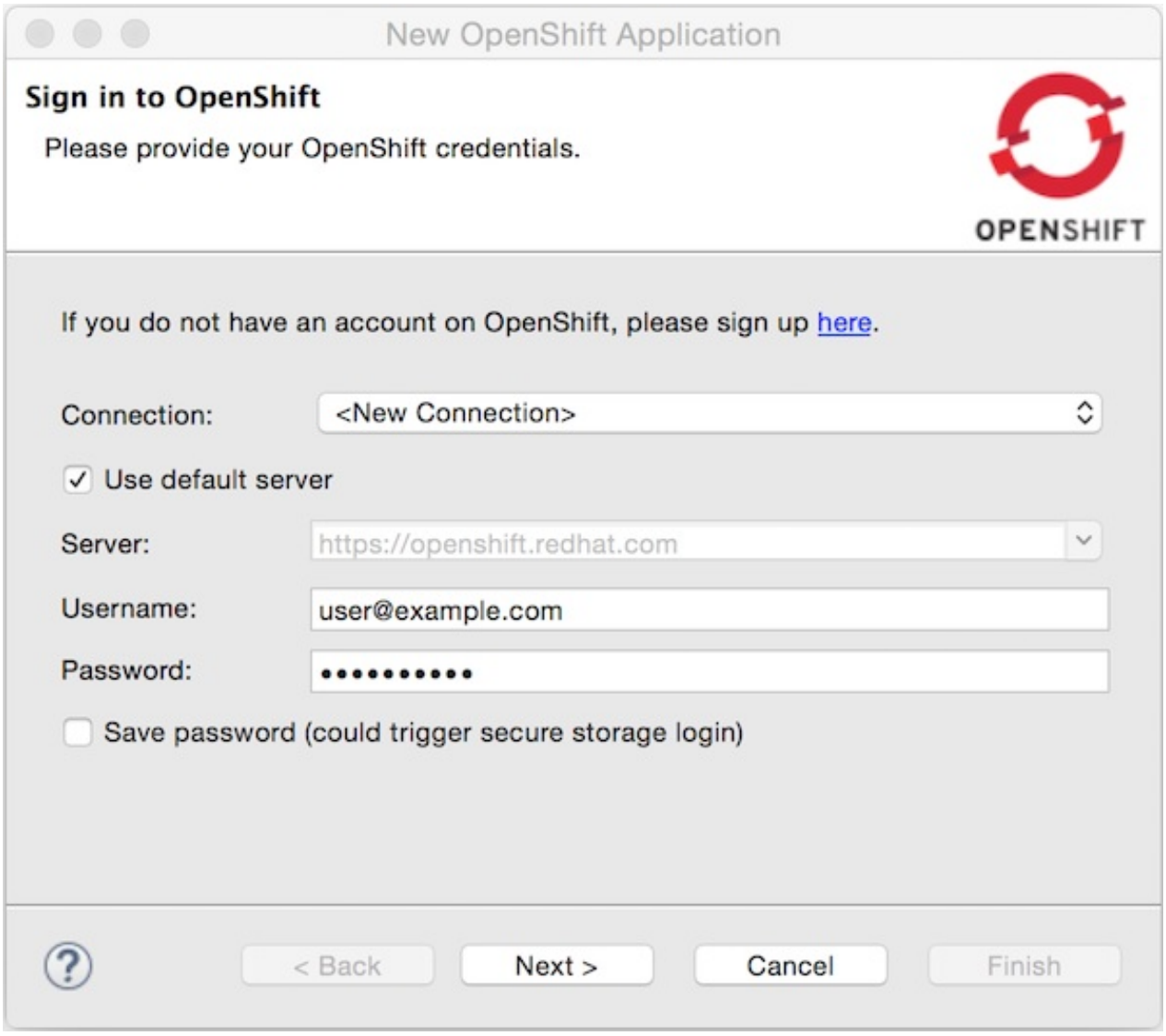

Figure 4.4. Completed Username and Password Fields

4. Click **Next**.
5. If your OpenShift Online account has no SSH public keys stored, you are prompted with the **Add SSH Keys** wizard. Click **New** and complete the fields about the SSH Keys to be created as follows:
   - In the **Name** field, type a name for the SSH key.
   - From the **Key Type** list, ensure **SSH_RSA** is selected.

- In the **SSH2 Home** field, ensure your **.ssh** directory path is shown.
- In the **Private Key File Name** field, type a name for the private key file name. The **Public Key File Name** field populates automatically with the name of the private key file name with **.pub** appended.

Click **Finish** and click **Finish** again to close the **Add SSH Keys** window.

6. If your OpenShift Online account has no domains, you are prompted with the **Create Domain** wizard. In the **Domain name** field, type a name for your new OpenShift Online domain and click **Finish**. The provided domain name must be unique across all domains on OpenShift Online; if it is not unique, you are directed back to the **Create Domain** wizard page to provide a unique domain name.
7. Complete the fields about the type of OpenShift application you want to create as follows:
   - Ensure **Create a new OpenShift application** is selected.
   - Expand **Basic Cartridges** and select **JBoss Enterprise Application Platform 6 jbosseap-6**.

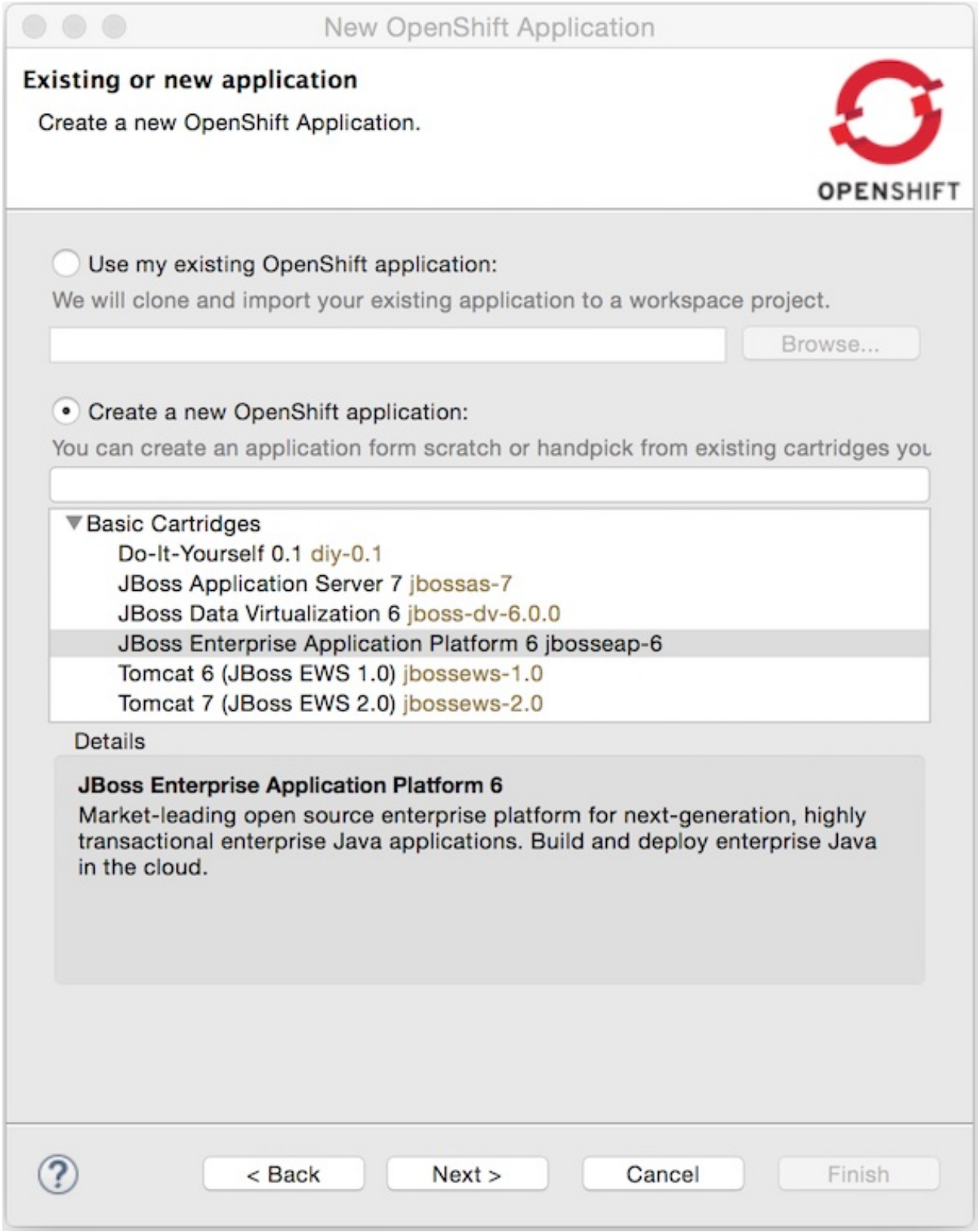

Figure 4.5. Completed Fields about the Type of OpenShift Application

8. Click **Next**.
9. Complete the fields about your OpenShift Online application as follows:
   - Ensure the **Domain** field displays the OpenShift Online domain with which you want to host your application.
   - In the **Name** field, type **ticketmonster**.

- From the **Gear profile** list, select **small**.

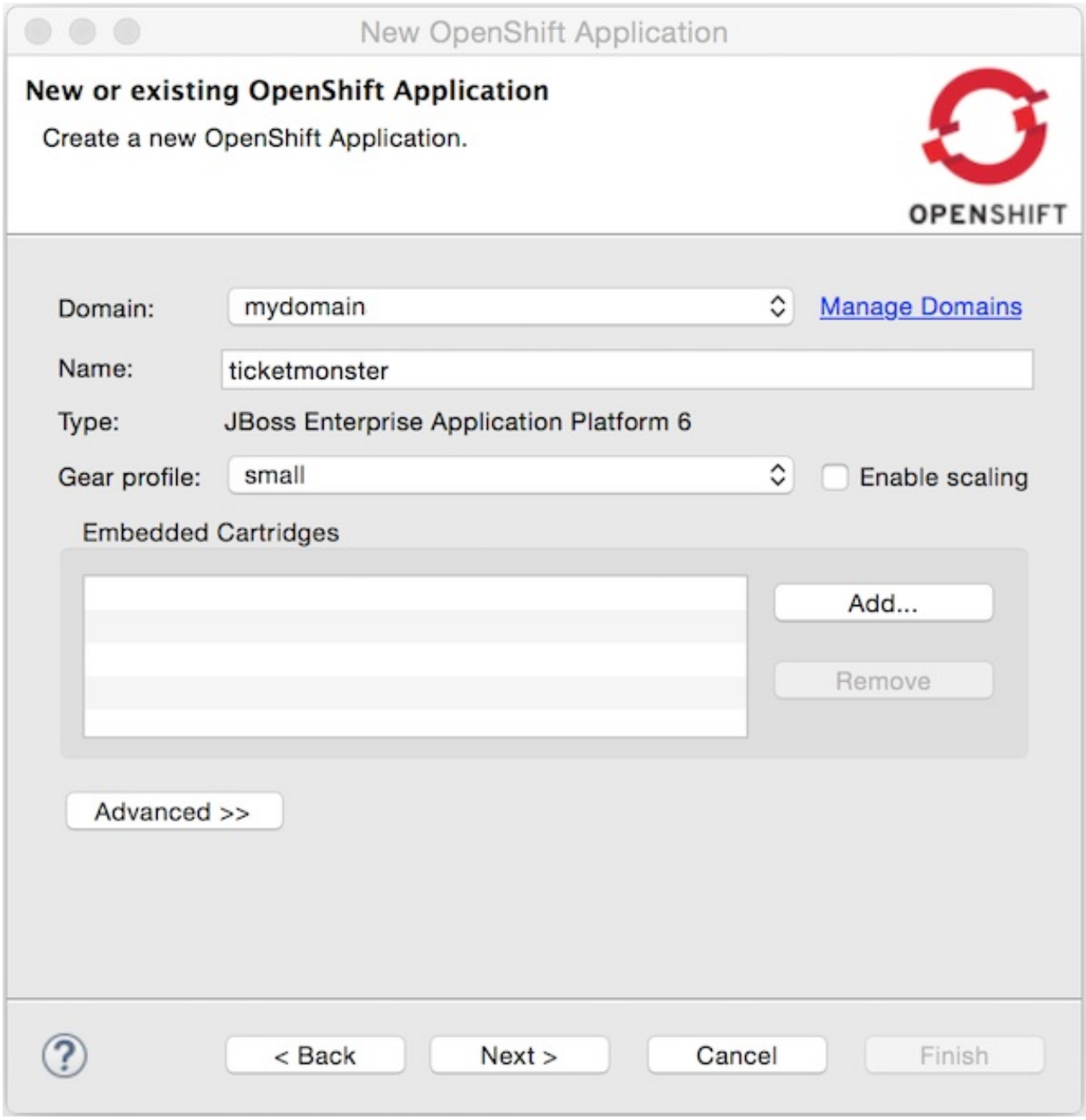

Figure 4.6. Completed Fields about the New OpenShift Application

10. Click **Next**.
11. Complete the fields about the workspace ticket-monster project as follows:
    - Clear the **Create a new project** check box.
    - In the **Use existing project** field, type **ticket-monster**. Alternatively, click **Browse** to select the **ticket-monster** project.
    - Ensure the **Create and set up a server adapter for easy publishing** check box is selected.

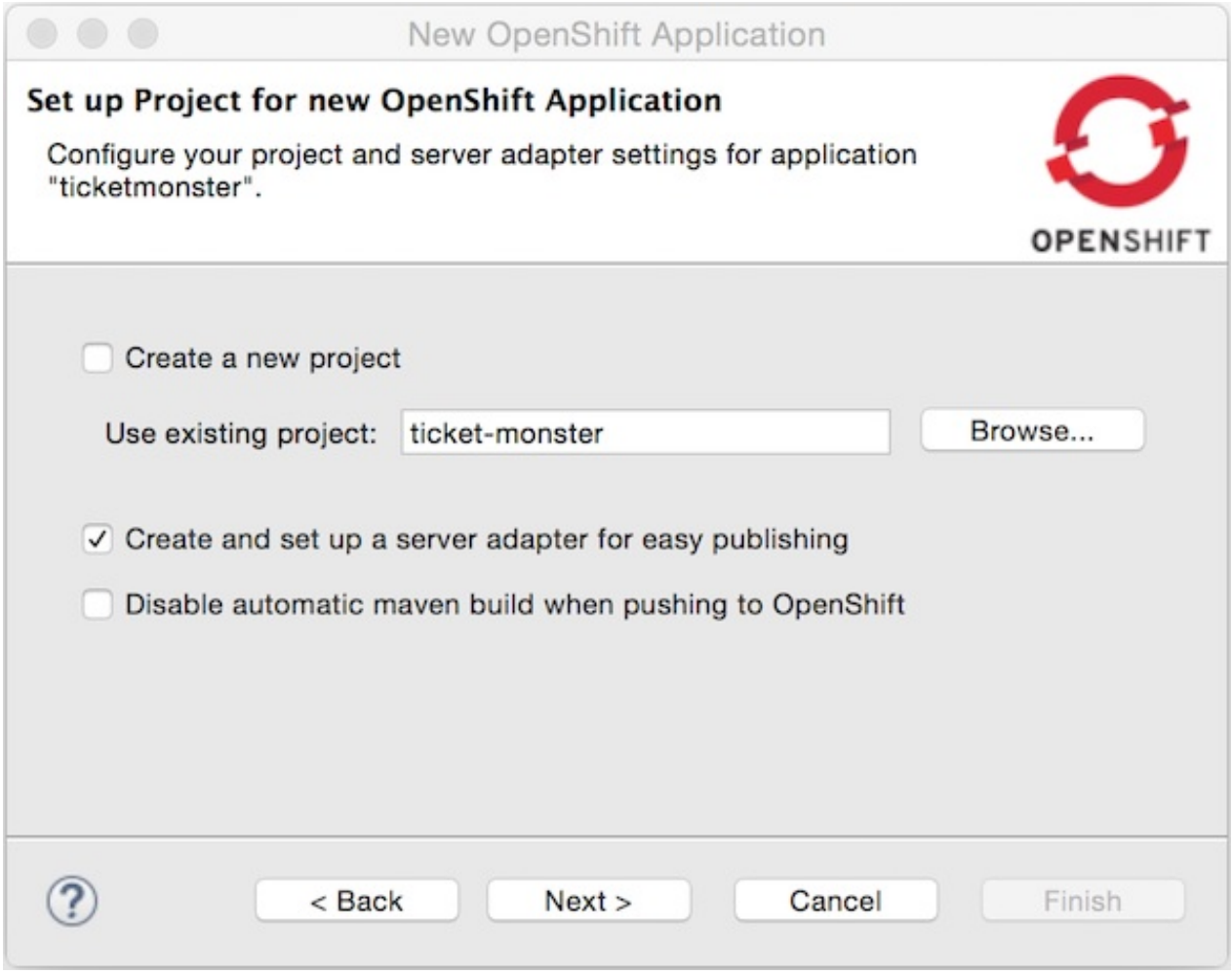

Figure 4.7. Completed Fields in the New OpenShift Application Wizard

12. Click **`Next`**.
13. Click **`Finish`** to create the new OpenShift application based on the existing workspace ticket-monster project. This process may take some time to complete.
14. At the prompt stating **`OpenShift application ticketmonster will be enabled on project ticket-monster ...`**, click **`OK`**. This configures the workspace ticket-monster project for OpenShift and connects it to the OpenShift Online Git repository system used for version control.

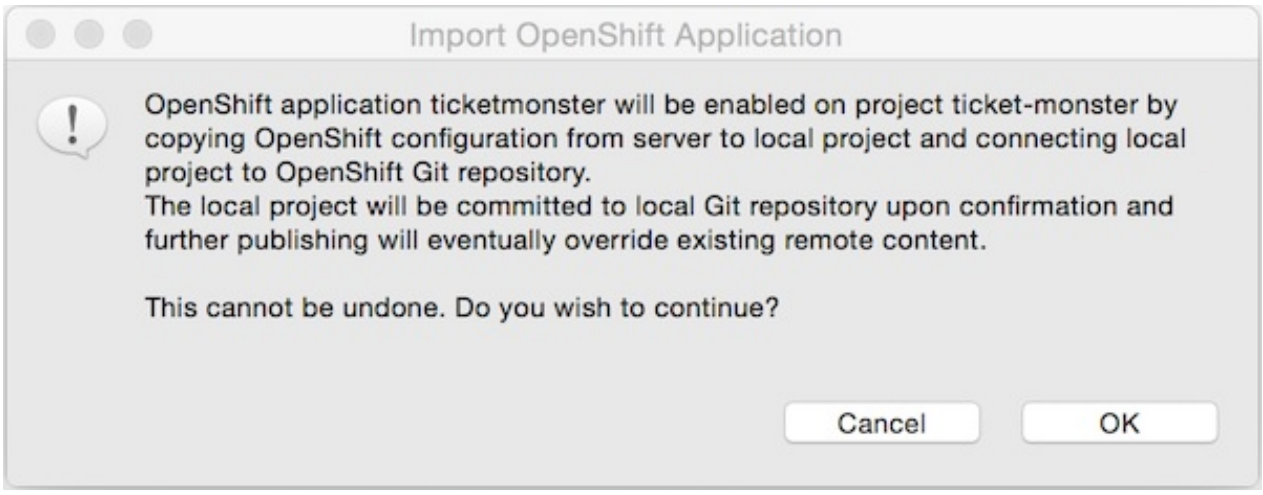

Figure 4.8. Import OpenShift Application Prompt

15. At the prompt stating the authenticity of the host cannot be established and asking if you are sure you want to continue connecting, verify the host information is correct and click **Yes**.
16. In the **Servers** view, right-click **ticketmonster at OpenShift** and click **Publish**.
17. At the prompt asking if you want to push committed changes to OpenShift, click **Yes**. The **Console** view automatically becomes the view in focus and displays the output from the OpenShift Online server. Once the OpenShift Online ticketmonster application is created and deployed, the **Console** view displays the following:

```
Deployment completed with status: success
```

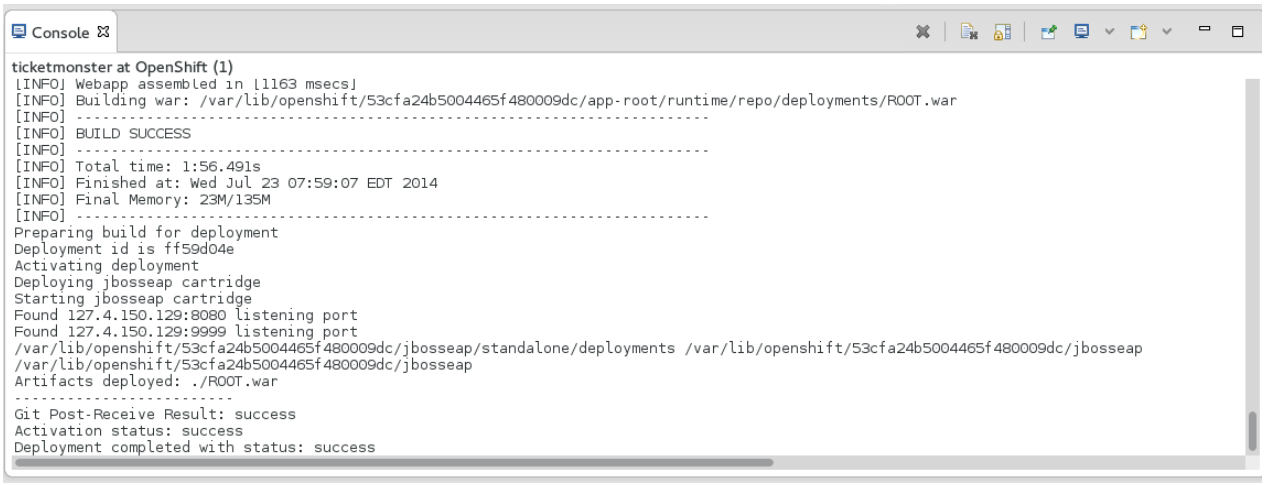

Figure 4.9. Deployment Completed Message in the Console View

Report a bug

4.5.3. View the OpenShift Application in a Browser

Once deployed, you can view the OpenShift Online ticketmonster application in a web browser. Details are provided here for viewing in your default system web browser and with BrowserSim.

To view the application in your default system web browser, in the **OpenShift Explorer** view, expand the connection and domain. Right-click **ticketmonster** and click **Show in Web Browser**.

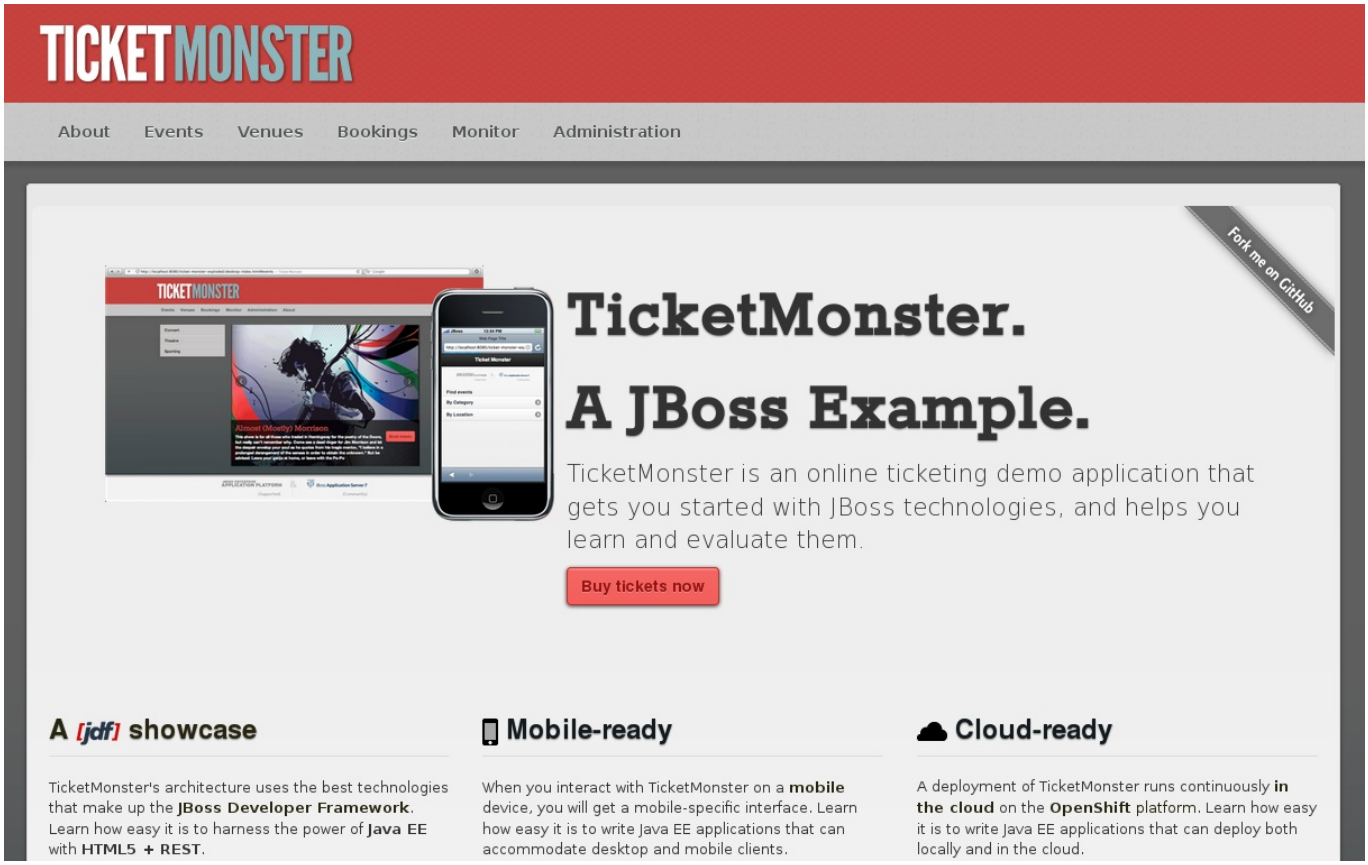

Figure 4.10. TicketMonster Application Opened in a Web Browser

To view the application in a BrowserSim simulated device web browser, click the **`Run BrowserSim`** icon. Copy and paste the URL from your default system web browser into the simulated device web browser address bar. Notice that the user interface is different in the two web browsers; the TicketMonster example has a different user interface for small views such as mobile devices.

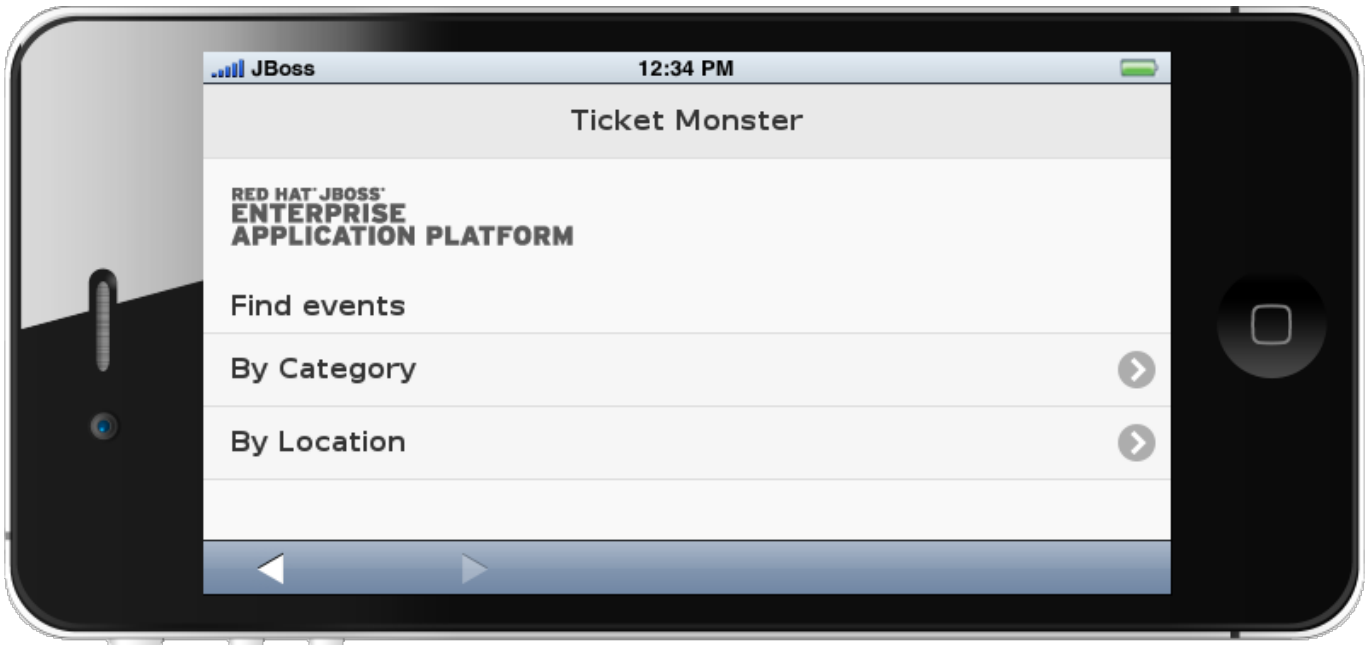

Figure 4.11. Ticket Monster Application Opened in BrowserSim

Report a bug

4.6. Change the Application

The project HTML resources can be edited with the JBoss Tools HTML Editor, as demonstrated in the procedure below.

Procedure 4.5. Change the Application

1. In the **Project Explorer** view, expand **ticket-monster** → **src** → **main** → **webapp** → **resources** → **templates** → **desktop**.
2. Double-click **home.html** to open it in the JBoss Tools HTML Editor.
3. Ensure the JBoss Tools HTML Editor **Source** tab is the tab in focus.
4. Locate and delete the following:

```
TicketMonster is an online ticketing demo application that gets you started with JBoss technologies, and helps you learn and evaluate them.
```

5. Replace it with:

```
TicketMonster is great!
```

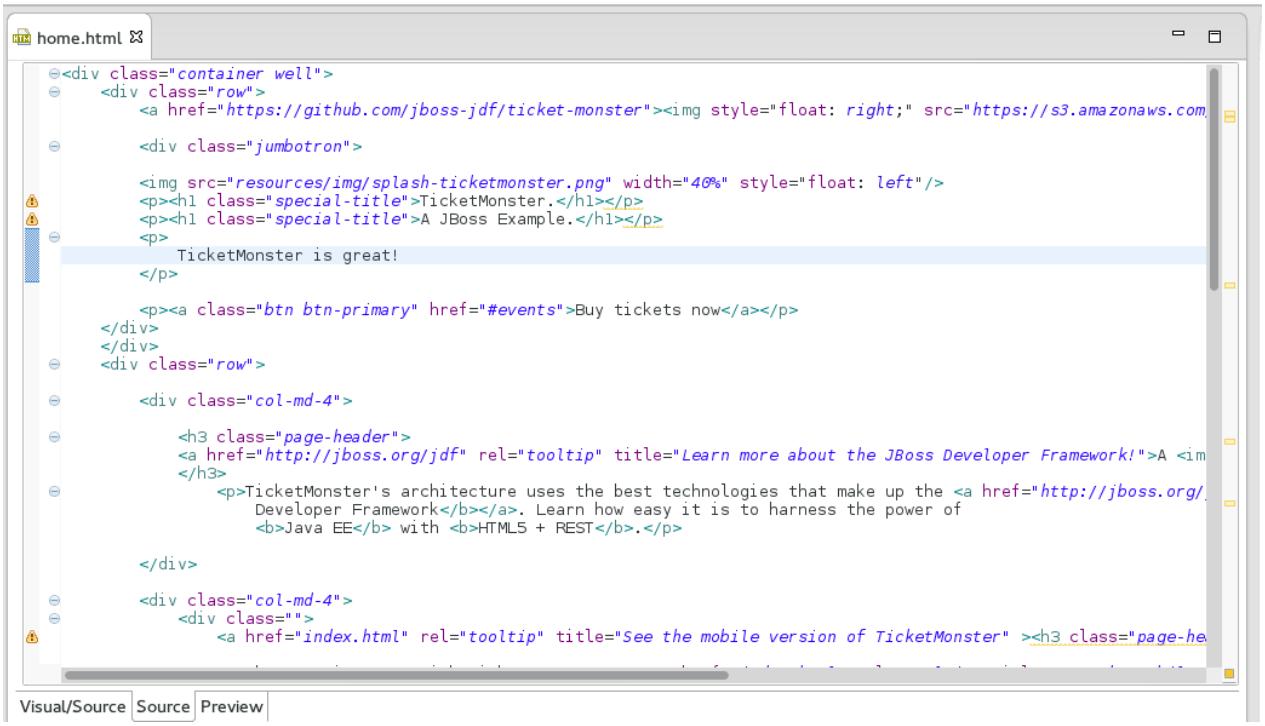

Figure 4.12. Text Replaced in the home.html File

6. Save the **home.html** file by pressing **Ctrl+S**. Alternatively, to save click **File** → **Save** or click the **Save** icon.

Note

In the **Project Explorer** view, notice that an arrowhead now precedes **home.html**, indicating that the local version of the file has been changed but those changes are yet to be committed to Git.

Report a bug

4.7. Redeploy the Changed Application to OpenShift

You must republish the application to OpenShift Online in order to see the **home.html** changes reflected in your default system web browser, as detailed here.

To republish the application, in the **Servers** view, right-click the **ticketmonster at OpenShift** server adapter and click **Publish**. At the prompt asking if you want to commit the project changes and push them to the OpenShift Online Git repository of the application during publishing, in the **Git Commit Message** text field type **Updated home.html** and click **Yes**. Once the application is rebuilt and redeployed, the **Console** view displays the following:

```
Deployment completed with status: success
```

Figure 4.13. Deployment Completed Message in the Console View

To view the changed application in your default system web browser, click the **Reload** icon. Alternatively, in the **OpenShift Explorer** view, expand the connection and domain. Right-click **ticketmonster** and click **Show in Web Browser**. The text is updated to reflect the changes made to **home.html**, indicating that you are viewing the updated running version of the application.

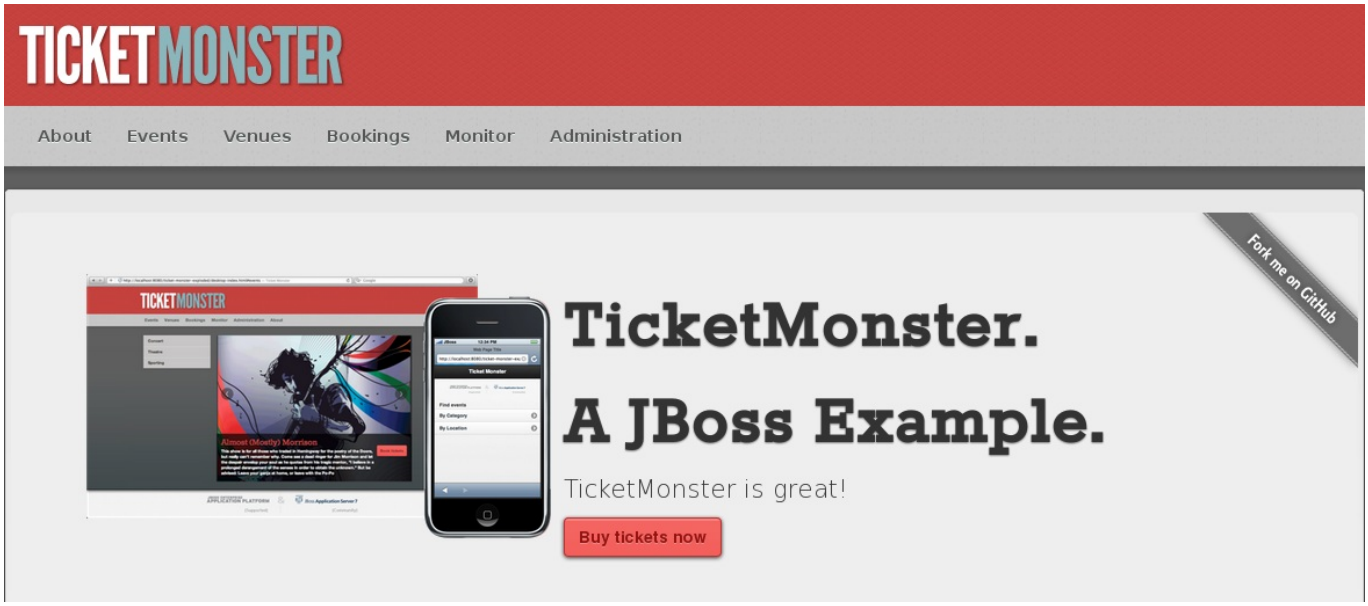

Figure 4.14. Updated TicketMonster Application

Report a bug

Chapter 5. Additional Tutorials

Now that you have completed the tutorials provided here, you may be looking for additional tutorials to learn more about JBoss Web Framework Kit and JBoss Developer Studio. Here is a list of some more tutorials.

Exploring the Examples

This guide examines the source code of a number of the JBoss Web Framework Kit examples and quickstarts. You can access the guide with the other JBoss Web Framework Kit documentation on the Red Hat Customer Portal.

Getting Started with Red Hat JBoss Developer Studio

This guide introduces new users to JBoss Developer Studio through a step-by-step tutorial. You can access the guide with the other JBoss Developer Studio documentation on the Red Hat Customer Portal.

TicketMonster Tutorial

This tutorial guides users in creating the moderately complex TicketMonster application. TicketMonster is an online ticket-booking application that allows users view and buy tickets for events. You can access the tutorial at http://www.jboss.org/ticket-monster/ on the JBoss Developer website.

Report a bug

Revision History

Revision 2.7.0-1	Tues Jan 06 2015	Michelle Murray
Generated for WFK 2.7 release		